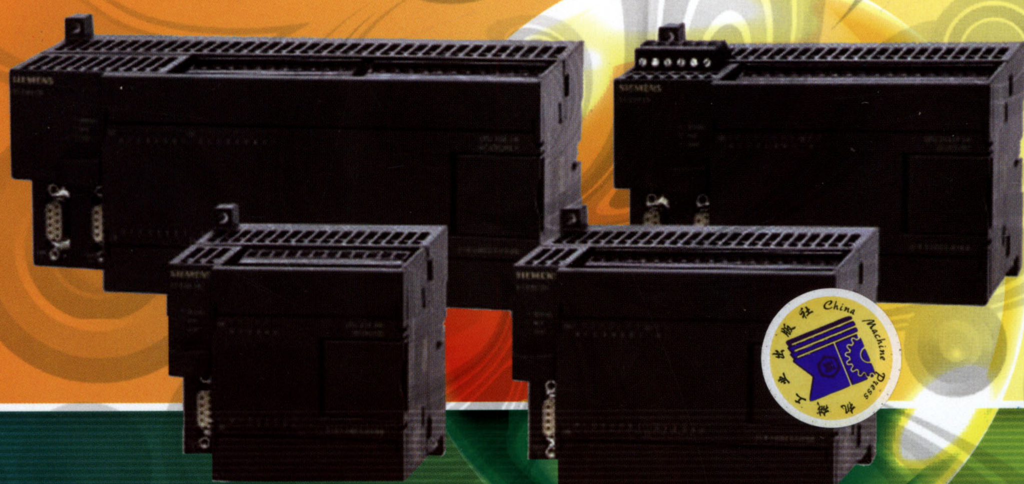


零基础轻松学会自动化技术丛书

零基础轻松学会

西门子S7－300／400

王时军　等编著







机械工业出版社
CHINA MACHINE PRESS

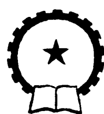
VISIT…

LANZAROTE Caliente.com

零基础轻松学会自动化技术丛书

零基础轻松学会西门子 S7-300/400

王时军　等编著



机 械 工 业 出 版 社

本书针对初学者，利用大量实例讲述西门子 S7-300/400 的编程与使用技术，以及西门子 S7-300/400 的基本和高级编程指令及程序调试、诊断；除此之外对 S7-300/400 的通信技术及软件的安装使用也有着详细图解。

本书是入门自学者的好帮手，也可作为大专院校相关专业师生、电气设计及调试编程人员的自学参考书。

图书在版编目（CIP）数据

零基础轻松学会西门子 S7-300/400/王时军等编著. —北京：机械工业出版社，2014.1

（零基础轻松学会自动化技术丛书）

ISBN 978-7-111-44821-1

Ⅰ. ①零…　Ⅱ. ①王…　Ⅲ. ①plc 技术　Ⅳ. ①TM571.6

中国版本图书馆 CIP 数据核字（2013）第 274551 号

机械工业出版社（北京市百万庄大街 22 号　邮政编码 100037）

策划编辑：朱　林　责任编辑：朱　林　版式设计：霍永明

责任校对：刘怡丹　封面设计：路恩中　责任印制：张　楠

北京玥实印刷有限公司印刷

2014 年 1 月第 1 版第 1 次印刷

184mm×260mm・13.75 印张・339 千字

0001—3000 册

标准书号：ISBN 978-7-111-44821-1

定价：39.80 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务　网络服务

社服务中心：（010）88361066　教 材 网：http://www.cmpedu.com

销 售 一 部：（010）68326294　机工官网：http://www.cmpbook.com

销 售 二 部：（010）88379649　机工官博：http://weibo.com/cmp1952

读者购书热线：（010）88379203　**封面无防伪标均为盗版**

前　　言

S7-300/400 是针对中小客户而设计的、可以自由组合的 *PLC* 产品。作为西门子公司可编程序控制器的主流产品，其市场占有率很高。以西门子 *S7-300/400* 为代表的 *PLC* 已成为我国工业控制领域中主要的工业自动化控制装置之一，为工业自动化提供了安全可靠和比较完善的解决方案。

本书从应用角度出发，简明扼要地介绍了西门子公司的 *S7-300/400* 系列 *PLC* 的基本结构，分析了 *S7-300/400 PLC* 设计开发方法，力求浅显易懂，使初学者容易入门。

本书首先介绍了 *S7* 编程软件的指令，并用实例方式介绍其应用；进而讲解程序的设计及块功能的编写；然后讲述其仿真软件的调试，并以实例方式讲述 *S7-300/400* 的网络通信及配置办法，旨在注重实际，强调应用。本书是一本工程性和实践性较强的书籍，可供企业工程技术人员作为培训教材使用，也可作为高等院校电气工程及其自动化、工业自动化、机电一体化、生产过程自动化、电力系统自动化、工业网络技术、电子信息工程、应用电子技术等专业的教材，同时对 *S7-300/400* 系列 *PLC* 的用户也有很大的参考价值。

本书由王时军老师编写第 *1～3* 章由李可德、李柄权、张舒、郭栋编写第 *4*、*5* 章，由林佟伟、杨家维、武鹏程编写第 *6～8* 章，并由王时军统稿。

由于编著者水平有限，书中难免有不足与纰漏之处，还望广大读者包涵，欢迎专业朋友指正！

编著者

目　　录

前言

第 1 章　SIMATIC S7-300/400 PLC 系统概述 …… 1

1.1　PLC 的基础概述 …… 1

1.1.1　PLC 概述 …… 1

1.1.2　PLC 的特点 …… 2

1.1.3　PLC 的组成 …… 2

1.2　SIMATIC 系列产品概述 …… 4

1.2.1　SIMATIC 可编程序控制器 …… 4

1.2.2　S7-300/400 产品特性 …… 5

第 2 章　S7-300/400 硬件及仿真软件安装 …… 7

2.1　安装 S7-300/400 …… 7

2.1.1　硬件需求 …… 7

2.1.2　安装 STEP7 …… 7

2.1.3　STEP 7 安装注意事项 …… 13

2.2　组态 …… 14

2.2.1　单机架或多机架的安装 …… 14

2.2.2　电气保护措施及接地 …… 15

2.3　安装 …… 16

2.3.1　安装导轨 …… 16

2.3.2　将模块安装在导轨上 …… 16

2.3.3　对模块贴标签 …… 17

2.4　接线 …… 18

2.4.1　保护接地导线和导轨的连接 …… 18

2.4.2　前连接器接线及插入 …… 18

2.4.3　安装屏蔽连接元件及屏蔽电缆 …… 19

2.5　寻址 …… 20

2.5.1　模块通道寻址方式 …… 20

2.5.2　寻址信号模块 …… 21

2.6　仿真软件的安装 …… 22

2.6.1　安装 PLCSIM …… 22

2.6.2　PLCSIM 与真实的 PLC 的差别 …… 22

第 3 章　S7-300/400 快速入门 …… 24

3.1　STEP 7 软件入门 …… 24

3.2　STEP 7 硬件组态 …… 26

3.2.1　创建一个项目 …… 26

3.2.2　硬件组态 …… 28

3.2.3　配置机架 …… 29

3.2.4　硬件更新 …… 33

3.3　在线调试 …… 36

3.3.1　设置 PG/PC 接口 …… 36

3.3.2　建立在线连接 …… 38

3.4　硬件调试与诊断 …… 42

3.4.1　硬件状态指示灯 …… 42

3.4.2　设置块测试环境 …… 44

3.5　控制和监视变量 …… 46

3.5.1　变量表 …… 46

3.5.2　修改变量 …… 50

3.6　测试程序 …… 52

3.6.1　监视程序状态 …… 52

3.6.2　断点调试 …… 55

第 4 章　S7-300/400 指令功能及应用 …… 58

4.1　PLC 程序概述 …… 58

4.1.1　程序的组成与结构 …… 58

4.1.2　变量编程及存储区 …… 59

4.1.3　指令符号及寻址方式 …… 64

4.2　位逻辑指令 …… 66

4.2.1　位逻辑运算指令 …… 66

4.2.2　比较指令 …… 68

4.3　定时器、计数器指令 …… 69

4.3.1　定时器指令 …… 69

4.3.2　计数器指令 …… 71

4.4　数据处理指令与逻辑控制指令 …… 72

4.4.1　数据处理指令 …… 72

4.4.2　数据传输指令 …… 73

4.4.3　状态字指令 …… 74

4.4.4　控制指令 …… 75

4.5　运算指令 …… 77

4.5.1　数学运算指令 …… 77

4.5.2　移位与循环指令 …… 79

4.5.3　字逻辑运算指令 …… 80

4.6　编程举例 …… 80

4.6.1　自保持（自锁）程序 …… 81
4.6.2　互锁程序 …… 81
4.6.3　延时程序 …… 81
4.6.4　分支程序 …… 82
第5章　S7-300/400的工程程序设计 …… 83
5.1　功能与功能块的应用 …… 83
5.1.1　S7-300的用户程序结构 …… 83
5.1.2　功能的生成与调用 …… 84
5.2　组织块的应用 …… 85
5.2.1　组织块与中断 …… 85
5.2.2　时间中断组织块 …… 88
5.2.3　硬件中断组织块 …… 98
5.2.4　延时中断组织块 …… 104
5.2.5　循环中断组织块 …… 109
5.2.6　同步循环中断组织块 …… 114
5.2.7　异步故障中断组织块 …… 115
第6章　顺序控制及S7 Graph编程 …… 117
6.1　顺序控制设计法 …… 117
6.1.1　顺序功能图 …… 117
6.1.2　顺序功能图结构 …… 118
6.2　S7 Graph顺序功能图编程 …… 120
6.2.1　认识S7 Graph编程环境 …… 120
6.2.2　了解S7 Graph调试方法 …… 129
6.2.3　分支程序的应用 …… 132
6.2.4　实例演示多个顺序器程序 …… 135
6.2.5　监控程序的应用 …… 141
6.2.6　功能块输入和输出参数 …… 144
第7章　S7-300/400的网络通信 …… 146
7.1　西门子PLC网络 …… 146
7.1.1　现代工业企业典型网络结构 …… 146
7.1.2　西门子PLC网络分类 …… 147
7.2　MPI通信概述 …… 149
7.2.1　MPI网络概述 …… 149
7.2.2　通信方式 …… 150
7.2.3　基本MPI的项目组态实例 …… 150
7.3　PROFIBUS的结构与硬件 …… 168
7.3.1　PROFIBUS概述 …… 168
7.3.2　基本PROFIBUS的项目实例 …… 169
第8章　S7-300/400 PLC在模拟量及闭环控制系统中的应用 …… 187
8.1　模拟量 …… 187
8.1.1　模拟量概述 …… 187
8.1.2　模拟量输入模块 …… 188
8.1.3　模拟量输出模块 …… 192
8.2　闭环控制 …… 197
8.2.1　闭环控制概述 …… 197
8.2.2　闭环控制功能块 …… 197
附录 …… 207
附录A　S7指令速查表 …… 207
附录B　梯形图指令速查表 …… 213

第 1 章

SIMATIC S7-300/400 PLC系统概述

1.1　PLC 的基础概述

1.1.1　PLC 概述

为工业设计而制造的可编程序控制器（Programmable Controller）是计算机家族的一员，随着应用的发展，为了能够代替继电器实现逻辑控制功能，遂诞生了可编程序逻辑控制器（Programmable Logic Controller），简称 PLC。

最早的 PLC 是在 1969 年，美国数字设备公司（DEC）为投标于美国通用汽车公司而设计制造了它。直到 1987 年，国际电工委员会（IEC）对 PLC 的定义做出如下归纳：PLC 是一种进行数字运算的电子系统，是专为在工业环境下的应用而设计的工业控制器。它采用可编程的存储器来存储指令，实现逻辑运算、顺序控制、定时、计数及算术运算等操作，并通过数字式或模拟式的输入和输出控制各种机械的生产过程。PLC 及其有关外部设备，都按易于与工业控制系统连成一个整体，易于扩充其功能的原则设计。

由此可见，PLC 是一种特殊的计算机系统，它既具有完成各种各样控制的功能，又具有和其他计算机通信联网的功能。PLC 以微处理器为基础，结合计算机技术、自动控制技术、通信技术，针对工业环境设计，可移植性高并且工作可靠。

从诞生之初到如今的 PLC，发展过程经历了 3 个阶段：从 20 世纪 70 年代至 80 年代中期，以单片机为主发展硬件技术，为取代传统的继电器、接触器控制系统而设计了各种 PLC 的基本型号；到 80 年代末期，为适应柔性制造系统（FMS）的发展，在提高单机功能的同时，加强软件的开发，提高通信能力；90 年代以来，为适应计算机集成制造系统（CIMS）的发展，采用多 CPU 的 PLC 系统，不断提高运算速度和数据处理能力，通信能力进一步提高。“网络就是计算机”这一观点已渗透到 PLC 领域，强大的网络通信功能更使 PLC 如虎添翼，随着各种功能模块、应用软件的开发，加速了 PLC 向连续控制、过程控制领域的发展。

PLC 的总发展趋势是向高集成度、小体积、大容量、高速度、易使用、高性能方向发展，一方面在与计算机的结合更紧密的情况下，制造出更多模块化的产品；另一方面，PLC 的网络与通信能力的增强，将会设计制造出更多标准化与多样化的产品，让工业软件在新的时代有着更迅速的发展。

1.1.2 PLC 的特点

PLC 如此迅速的发展，一方面基于工业自动化的客观需要，另一方面它有许多独特的优点。比如：能较好地解决工业控制领域中用户普遍关心的可靠、安全、灵活、方便、经济等，PLC 的特点介绍如下：

1. 具有高可靠性

对 PLC 选用的器件进行了严格筛选，PLC 的输入输出电路均采用光电隔离技术，屏蔽工业现场的干扰信号。在输入电路中还普遍采用 RC 滤波。PLC 的 CPU 具有自诊断功能，能对异常情况进行有效处理。大型 PLC 还可以采用由双 CPU 构成冗余系统或由三 CPU 构成表决系统，使可靠性进一步提高。

2. 具有丰富的模块

工业现场的信号有很多种，比如交流和直流信号、开关量和模拟量信号、电压和电流信号等。PLC 针对不同的工业现场信号，设计了相应的信号处理模块与工业现场的器件或设备进行连接。现代的 PLC 在人—机接口模块、通信模块方面有更强的功能。

3. 采用模块化结构

为了适应各种工业控制需要，除了单元式的小型 PLC 以外，绝大多数 PLC 都采用模块化结构，PLC 的各个部件划分为 CPU、电源、I/O 等多种模块，由机架及电缆将各模块连接起来，系统的规模和功能可根据用户的需要自行组合。

4. 编程简单易学

PLC 的编程一般都支持梯形图语言，它类似于继电器控制线路，对使用者来说不需要具备计算机的专门知识，因此很容易被一般工程技术人员所理解和掌握。

5. 安装简单，维修方便

PLC 可以在各种工业环境下直接运行，使用时只需将现场的各种设备与 PLC 相应的 I/O 端相连接即可投入运行，各种模块上均有运行和故障指示装置，便于用户了解运行情况和查找故障。模块化结构的系统，一旦某模块发生故障，用户可以通过更换模块的方法使系统迅速恢复运行。

1.1.3 PLC 的组成

PLC 之所以实现控制，就是按一定算法进行输入/输出（I/O）变换，并将这个变换予以物理实现。说简单点，就是按一条条的指令，不断地在输入/输出间存储转换。

由此我们可以说 PLC 是微型计算机技术与机电控制技术相结合的产物，是一种以微处理器为核心，用于电气控制的特殊计算机，它采用典型计算机结构，主要由中央处理器（Central Processing Unit，CPU）、存储器、I/O 接口、电源、通信接口、扩展接口等单元部件组成，这些单元部件都是通过内部总线进行连接的，如图 1-1 所示。

1. 中央处理器

PLC 的中央处理器与普通计算机控制系统一样，一般由控制器、运算器、寄存器等组成。CPU 是 PLC 的核心，一切逻辑运算及判断都是由它完成的，并控制所有其他部件的操作。CPU 通过数据总线、地址总线和控制总线与存储器、I/O 接口电路等相连接。用户程序和数据存放在存储器中，当 PLC 处于运行方式时，CPU 按扫描方式工作，从用户程序第一

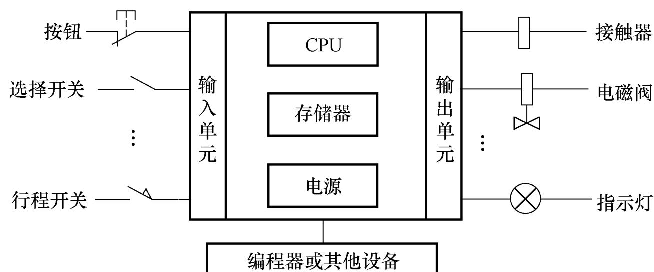


图 1-1 PLC 的基本结构

条指令开始，直至用户程序的最后一条指令，不停地周期性扫描，每扫描完成一次，用户程序就执行一次。

CPU 的主要功能有从存储器中读指令、执行指令、处理中断等。

2. 存储器

存储器是具有记忆功能的半导体集成电路，用来存储系统程序、用户程序、逻辑变量以及数据、系统组态和其他一些信息。

系统程序是用来控制和完成 PLC 各种功能的程序，这些程序是由 PLC 制造商用相应 CPU 的指令系统编写的，并固化到 ROM（只读存储器）中。

用户程序存储器用来存放由编程器或其他设备输入的用户程序。用户程序由使用者根据工程现场的生产过程和工艺要求而编写，可通过编程器或编程软件修改。在 PLC 中使用两种类型的存储器：一种是只读类型的存储器，如 ROM、PROM、EPROM 和 E^2PROM 等；另一种是可读写的 RAM。现说明如下：

1）只读存储器。只读存储器可以用来存放系统程序，PLC 去电后再加电，系统程序内容不变且重新执行。只读存储器也可用来固化用户程序和一些重要的参数，以免因为偶然操作失误而造成程序和数据的破坏和丢失。

2）随机存储器。RAM 中一般存放用户程序和系统参数。当 PLC 处于编程工作方式时，用编程器或编程软件下载到 PLC 中的程序和参数存放到 RAM 中，当切换到运行方式时，CPU 从 RAM 中取指令并执行。用户程序执行过程中产生的中间结果也在 RAM 中暂时存放。RAM 通常为 CMOS 型集成电路，功耗小，速度快，但断电时内容丢失。所以在有的 PLC 中使用大电容或后备电池保证掉电后 PLC 中的内容在一定时间内不会丢失。

3. I/O 单元

I/O 单元又称为 I/O 模块，它是 PLC 与工业生产设备或工业过程连接的接口。现场的输入信号，如按钮、行程开关、限位开关以及各传感器输出的开关量或模拟量等，都要通过输入模块送到 PLC 中。由于这些信号电平各式各样，而 PLC 的 CPU 所处理的信息只能是标准电平，所以输入模块还需要将这些信号转换成 CPU 能够接收和处理的数字信号。输出模块的作用是接收 CPU 处理过的数字信号，并把它转换成现场的执行部件所能接收的控制信号，以驱动负载，如电磁阀、电动机、灯光显示等。

PLC 的 I/O 模块上通常都有接线端子，PLC 类型不同，I/O 模块的接线方式也不同，通常按其接线方式分为汇点式、分组式和隔离式这 3 种，如图 1-2 所示。

每个 I/O 模块分别只有一个公共端（COM）的被称为汇点式，其输入或输出点共用一

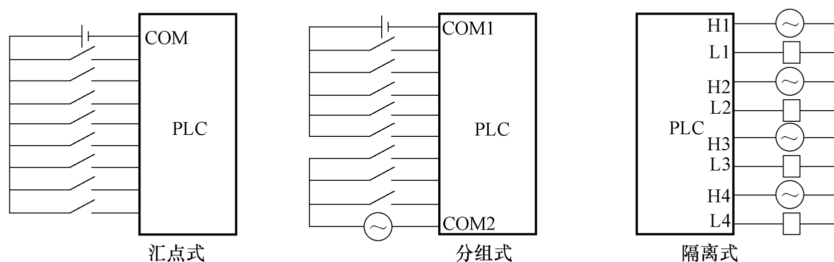


图 1-2　I/O 模块的 3 种接线方式

个电源；分组式是指将 I/O 端子分为若干组，每组的 I/O 电路有一个公共端并共用一个电源，组与组之间的电路隔开；隔离式是指具有公共端的各组 I/O 点之间互相隔离，可各自使用独立的电源。

4. 电源单元

PLC 的电源单元通常会将 220V 的单相交流电源转换成 CPU、存储器等电路工作所需的直流电，它是整个 PLC 系统的能源供给中心，电源的好坏直接影响 PLC 的稳定性和可靠性。

对于小型整体式 PLC，其内部有一个高质量的开关稳压电源，为 CPU、存储器、I/O 模块提供 5V 直流电源，还可为外部输入单元提供 24V 的直流电源。

5. 通信接口

为了实现微机与 PLC、PLC 与 PLC 间的对话，PLC 配有多种通信接口，如打印机、上位计算机和编程器等接口。

6. I/O 扩展接口

I/O 扩展接口用于将扩展单元或特殊功能单元与基本单元相连，使 PLC 的配置更加灵活，以满足不同控制系统的要求。

1.2　SIMATIC 系列产品概述

1.2.1　SIMATIC 可编程序控制器

据美国 ARC（Automation Research Corp）的调查，在全球 PLC 制造商中，前五大 PLC 生产厂家分别为西门子公司、Allen-Bradley（A-B）公司、施耐德（Schneider）公司、三菱（Mitsubishi）公司、欧姆龙（Omron）公司，他们的销售额约占全球总销售额的 2/3。

S7-200 是西门子公司的产品之一，其注册商标为 SIMATIC。西门子公司的产品最早是 1975 年投放市场的 SIMATIC S3，它实际上是带有简单操作接口的二进制控制器。在 1979 年，S3 系统被 SIMATIC S5 所取代，该系统广泛使用微处理器。20 世纪 80 年代初，S5 系统进一步升级产生 U 系列 PLC，较常用的机型有 S5-90U、95U、100U、115U、135U、155U。1994 年 4 月，S7 系列诞生，它具有更国际化、更高性能等级、安装空间更小、更良好的 Windows 用户界面等优势，它包括小型 PLC S7-200、中型 PLC S7-300 和大型 PLC S7-400。1996 年，在过程控制领域，西门子公司又提出 PCS7（过程控制系统 7）的概念，将其具有

优势的WINCC（与Windows兼容的操作界面）、PROFIBUS（工业现场总线）、COROS（监控系统）、SINEC（西门子工业网络）及控制技术融为一体。现在，西门子公司又提出TIA（Totally Integrated Automation）概念，即全集成自动化系统，将PLC技术融于全部自动化领域。

西门子公司的PLC产品有SIMATIC S7、M7和C7等几大系列。

SIMATIC S7 系列PLC是德国西门子公司在S5系列PLC基础上于1995年陆续推出的性价比较高的PLC系统。其中，微型的有SIMATIC S7-200系列，最小配置为8DI/6DO，可扩展2~7个模块，最大I/O点数为64 DI/DO、12 AI/4 AO；中型的有SIMATIC S7-300系列，最多可以扩展32个模块；中高档性能的有S7-400系列，可以扩展300多个模块。SIMATIC S7系列PLC都采用了模块化、无排风扇结构，且具有易于用户掌握等特点，使得S7系列PLC成为各种从小规模到中等性能要求以及大规模应用的首选产品。S7-300/400可以组成MPI、PROFIBUS和工业以太网等。

M7-300/400采用与S7-300/400相同的结构，它可以作为CPU或功能模块使用。其显著特点是具有AT兼容计算机功能，使用S7-300/400的编程软件STEP 7和可选的M7软件包，可以用C、C++或CFC（连续功能图）等语言来编程。M7适合于需要处理数据量大，对数据管理、显示和实时性有较高要求的系统使用。

C7由S7-300 PLC、HMI（人机接口）操作面板、I/O、通信和过程监控系统组成。整个控制系统结构紧凑，面向用户配置/编程、数据管理与通信集成于一体，具有很高的性价比。WinAC基于Windows操作系统和标准的接口（ActiveX，OPC），提供软件PLC或插槽PLC。WinAC基本型用于常规控制系统；WinAC实时型用于实时性、确定性要求非常高的控制场合（比如运动控制和快速控制等）；WinAC插槽型具有硬件PLC的所有特性，适用于实时性、安全性、可靠性要求较高的场合。

1.2.2 S7-300/400产品特性

S7-300/400是如今比较流行的PLC产品，其产品特点分别概述如下：

1. S7-300的产品特性

S7-300是西门子公司生产的中型PLC。产品采用了模块式结构，可控制的I/O点多、功能强、扩展性能好、使用灵活方便，既能用于机电设备的单机控制，又能组成大中型PLC网络系统，可满足工业自动化控制领域的绝大多数控制要求。产品的主要特点如下。

1）运算速度快。S7-300 PLC采用了先进的高速处理器，基本逻辑指令的执行时间最快可达0.05μs。

2）软件功能强。S7-300 PLC可用于复杂功能的编程与控制，且可采用STEP7、STEP7-Lite等编程软件，使用多种编程语言。

3）通信性能好。CPU模块集成有1~2个标准通信接口，支持MPI多点通信，可方便地连接编程器、文本显示器、触摸屏等外设。MPI还兼容了PROFIBUS-DP功能，部分CPU模块能够同时连接2条PROFIBUS-DP总线，以构成PLC网络系统。

4）适用范围广。S7-300 PLC不仅有规格众多的I/O扩展模块，而且还有众多的网络连接、模拟量I/O、温度测量、定位控制模块可供选择，可以用于各种不同的控制场合。

2. S7-400 的产品特性

S7-400 PLC 是目前西门子公司功能全、性能好、规格大、I/O 点数多的大型 PLC 产品，可以适用于各种大型复杂系统控制。分别体现在：

1）功能强大。可以安装多个 CPU 模块组成多 CPU、安全型、冗余控制系统。

2）扩展性能好。PLC 可控制的 I/O 点可达 262144 点，可构成大规模控制系统。

3）通信能力强。CPU 模块集成接口可达 4 个，便于构成大型分布式系统与网络系统。

4）运算速度快。基本逻辑指令的执行时间可达 0.03μs，可用于高速处理。

5）兼容性好。PLC 可与西门子公司早期的 S5-155U、S5-135U、S5-115 兼容，通过 S5 扩展接口，实现对 S5 系列 PLC 的控制。

第 2 章

S7-300/400硬件及仿真软件安装

2.1 安装 S7-300/400

2.1.1 硬件需求

STEP7 软件是用于 SIMATIC S7-300/400 PLC、M7 工业控制系统、C7 集成式 PLC 的标准工具软件。它由一系列应用程序组成，可根据需要在 STEP7 Basis 标准组件的基础上，选择扩展软件包增强功能。STEP7 Basis V5.2 标准版（Standard Package）的功能组成如图 2-1 所示，最低硬件配置如下所示：

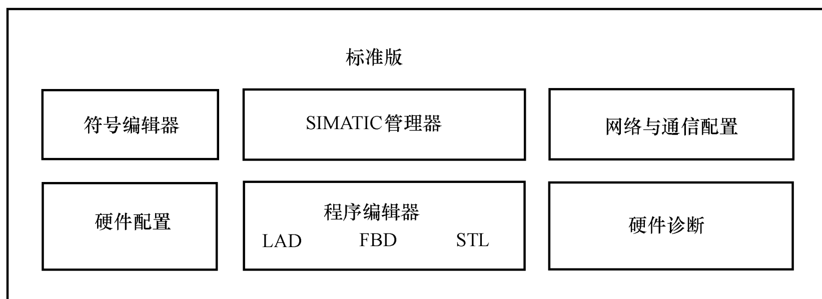


图 2-1　STEP7 Basis 标准功能

STEP7 软件可在 PC 上直接安装，不同版本的 STEP7 对计算机的硬件要求有所不同，STEP7 Basis V5.3 标准版的最低配置如下。

1）基本性能：CPU 速度 600 MHz 以上，内存大于 256 MB，硬盘空间不低于 300 MB。

2）操作系统：Microsoft Windows 2000 SP3 版以上或 Microsoft Windows XP Professional SP1.3 版以上。

2.1.2 安装 STEP7

1. 软件安装

将 STEP7 安装光盘插入 CD/DVD ROM 后，打开光盘，双击“Setup.exe”图标，如图 2-2所示。

可执行 STEP7 的自动安装程序，安装可根据弹出的引导信息按步骤进行，如图 2-3 所示。



图 2-2　打开安装文件

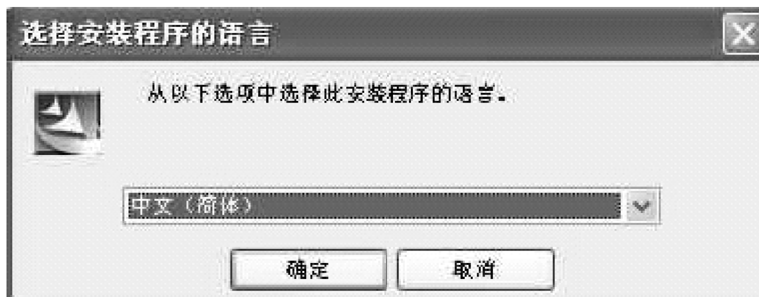


图 2-3　安装程序语言选择对话框

选择了语言（此处我们选择简体中文）之后，出现安装程序确认对话框，如图 2-4 所示。



图 2-4　安装确认对话框

单击“下一步”之后，将会出现提示阅读版权的信息，继续单击“下一步”，在出现确认版权及用户信息后，将出现询问安装方式对话框，如图 2-5 所示。

STEP7 软件可选择 3 种安装方式。

1）典型的安装：可以安装 STEP7 的所有语言、应用程序、项目示例和文档（一般选择此选项）。

2）最小安装：只安装一种语言和基本的 STEP7 程序。

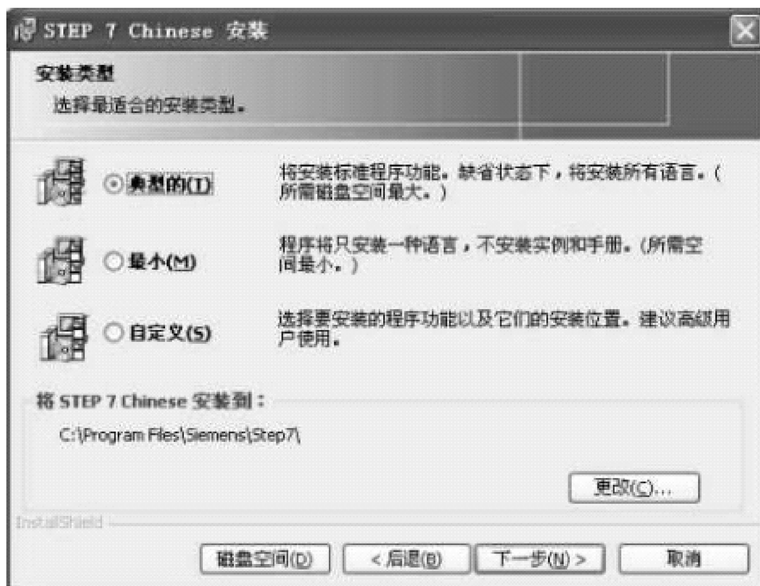


图 2-5　安装类型对话框

3）自定义安装：可选择 STEP7 安装的语言、程序、示例和文档。

STEP7 安装过程中，安装程序将自动检查计算机硬盘上是否有西门子公司的授权（License Key）。如果计算机没有安装授权，安装程序会自动提示用户进行授权的安装。授权的安装可以选择两种方式进行：在安装过程中直接安装授权；在使用时或稍后再安装授权。

此处我们选择"否，以后再传送许可证密钥"，再单击"下一步"，如图 2-6 所示。

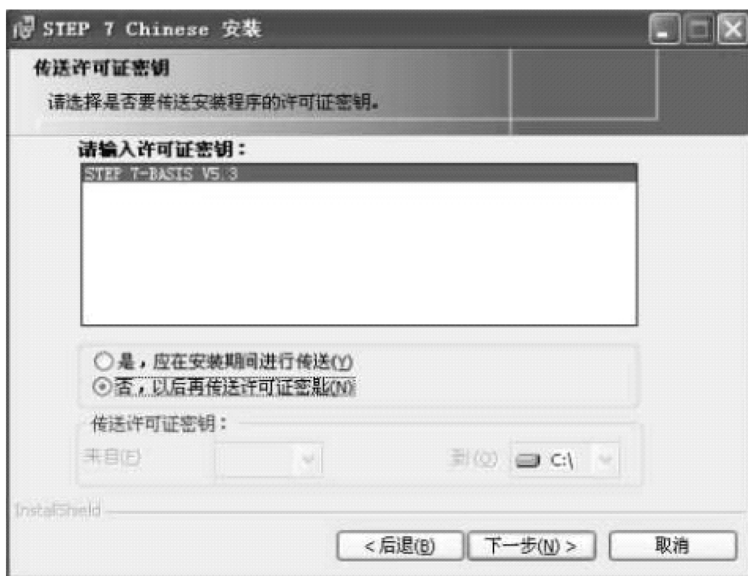


图 2-6　许可证密钥安装选择对话框


注意:

每安装一个授权，授权盘上的计数器将被减 1，当计数值为 0 时，就不能再使用这张磁盘进行授权的安装。使用 Authors W 程序可以把授权收回到授权盘，也可在硬盘的不同区之间移动、转移授权。

在图 2-7 所示的安装向导中，选择“存储卡参数分配”（本例选择“无”），然后单击“确认”→“下一步”继续安装。



图 2-7　存储卡安装对话框


注意:

存储卡选项含义

“无”：不安装 EPROM 驱动程序。

“内部编程设备接口”：需要选择适用于 PG（西门子专用编程设备）的条目安装。

“外部编程设备”：使用 PC（个人计算机），请选择外部编程器的驱动程序。在此，必须指定连接此编程器的端口（例如 LPT1）。

通过在 STEP 7 程序组或控制面板中调用“存储卡参数分配”程序，可以在安装后修改设定的参数。

在图 2-8 所示的安装向导中，选择通信方式（本例选择“PC Adapter”），然后单击“安装”，将自动弹出警告性的确认对话框，如图 2-9 所示，要求用户单击“是”，否则单击“否”重新选择。

在图 2-10 所示的安装向导中，单击“确定”继续安装，进行通信配置。


注意:

编程计算机的通信端口可通过计算机本身的操作系统，在“设备管理器”上设置，要确保编程计算机没有与其他通信发生中断冲突和地址重叠现象。



图 2-8　接口选择对话框

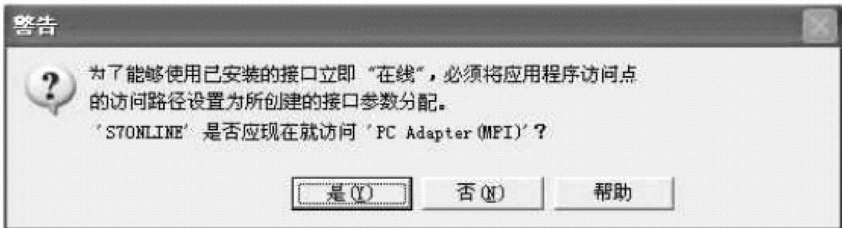


图 2-9　警告确认对话框



图 2-10　通信配置对话框

在图 2-11 所示的安装向导中，由于 Windows 系统激活了防火墙，要求允许 Windows 防火墙准许进入，单击“是”继续安装。

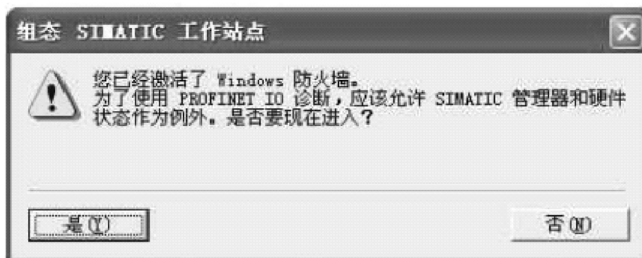


图 2-11　防火墙进入许可对话框

在图 2-12 所示的安装向导中，由于还要继续安装，单击“否”，等待全部安装完毕后再重新启动（热启动）PC。

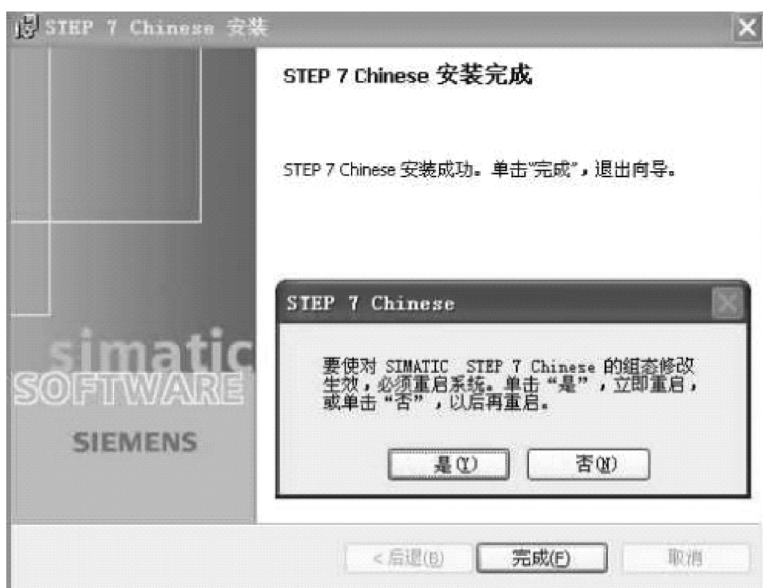


图 2-12　安装完成对话框

安装过程中，会出现 PG/PC 接口设置提示对话框，以定义 PG/PC 与 PLC 之间的通信接口。也可单击提示对话框“Cancel”忽略该步骤。在软件安装完成后，在计算机上通过“控制面板”中的“Setting PG/PC Interface”进行通信连接设置（参见后述）。

STEP7 安装结束时，会出现“存储卡配置参数（Memory Card Parameter Assignment）”设置对话框，提示用户为存储卡配置参数。也可单击提示对话框“Cancel”忽略该步骤。在软件安装完成后，根据需要，通过计算机“控制面板”中的“Memory Card Parameter Assignment”进行存储卡参数的设定。

软件安装完成后，通过单击计算机“开始”菜单中的“SIMATIC”→“SIMATIC Manager”选项，或双击桌面快捷方式图标，即可快速启动 STEP7 管理器。

在“STEP 7 V5.3”文件夹里，如图 2-2 所示，双击“S7-Plcsim”→“Disk1”→“Setup. exe”，安装 S7-Plcsim（S7 程序仿真器），双击“S7 Graph”→“Disk1”→“Setup. exe”，安装 S7 Graph 编程语言。

2. 安装授权

在此版本中，授权管理模块名为自动化许可证管理器，它是 Siemens AG 的软件产品，用于管理所有系统的许可证密钥（许可证模块）。要使用 STEP 7 编程软件，需要一个产品专用的许可证密钥（用户权限）。从 STEP 7 V5. 3 版本起，密钥可以通过自动化许可证管理器来安装。

合法使用受许可证保护的 STEP7 程序软件包时必须要有许可证。许可证为用户提供使用产品的合法权限。下列各项提供使用权限证明：CoL（许可证证书）和许可证密钥。

软件的实际特性取决于所安装的许可证密钥类型，西门子 AG 软件产品提供如表 2-1 所示的不同类型的面向应用的用户许可证。自动化许可证管理器通过 MSI 设置过程安装。STEP 7 产品 CD 包含自动化许可证管理器的安装软件。可以在安装 STEP 7 的同时安装自动化许可证管理器或在以后安装。

表 2-1　西门子 AG 软件许可证类型

许可证类型	说　　明
单独许可证	该软件可在无限制使用时间的单台计算机上使用
浮动许可证	该软件可在无限制使用时间的计算机网络(远程使用)上使用
试用许可证	该软件可在下列限制条件下使用 1)有效期最多为 14 天 2)第一次使用之日起的总操作天数 3)用于测试和确认(免除责任)
升级许可证	在软件升级方面,现有系统中的特定要求可能适用 1)升级许可证可用于将“旧版本 X”软件转换为新版本 X + 2)由于给定系统中需处理的数据量增大,可能需要升级

此处我们选择单独许可证的安装，其安装方法为：

在“STEP 7 V5. 3”文件夹里双击“Automation License Manager”→“Disk1”→“Setup. exe”，安装自动化许可证管理器。


注意：

在自动化许可证管理器的在线帮助以及在安装 CD-ROM 中 STEP 7Readme. wri 文件中有关于处理许可证密钥的信息。如果不遵守这些指南，可能会丢失许可证密钥且不可恢复。

2. 1. 3　STEP 7 安装注意事项

1. 安装环境要求

为保证 PLC 工作的可靠性，尽可能地延长其使用寿命。在安装时一定要注意周围的环境，其安装场合应该满足以下几点。

1）环境温度：工作时，在 0 ~ 55℃范围内；保存时，在 −20 ~ 70℃范围内。

2）环境相对湿度：35% ~ 85% RH（不结露）范围内，对于 Q 系列 PLC 为 5% ~ 95% RH-2 级。

3）不能受太阳光直接照射或水的溅射。

4）周围无腐蚀和易燃的气体，例如氯化氢、硫化氢等。

5）周围无大量的金属微粒、灰尘、导电粉尘、油雾、烟雾、盐雾等。

6）避免频繁或连续的振动：直接用螺钉安装，保证振动频率范围为 57～150Hz，$9.8m/s^2$；DIN 导轨安装时，保证振动频率范围为 57～150Hz，$4.9m/s^2$。

7）不能超过 15g（重力加速度）的冲击。

8）耐干扰能力：1000V（峰峰值），1ms 幅度，30～100Hz。

2. 安装注意事项

除满足以上环境条件外，安装时还应注意以下几点。

1）PLC 的所有单元必须在断电时安装和拆卸。

2）为防止静电对 PLC 组件的影响，在接触 PLC 前，先用手接触某一接地的金属物体，以释放人体所带静电。

3）注意 PLC 机体周围的通风和散热条件，切勿使导线头、铁屑等杂物通过通风窗落入机体内。

2.2 组态

2.2.1 单机架或多机架的安装

S7-300/400 的安装顺序步骤如图 2-13 所示。

按照图 2-13 的顺序，我们以 S7-300 安装为例，讲述其硬件组态。“组态”指的是在站配置机架（HW Config）窗口中对机架、模块、分布式 I/O（DP）机架以及接口子模块等进行排列。使用组态表示机架，就像实际的机架一样，可在其中插入特定数目的模块。

S7-300 PLC 由一个中央处理单元（CPU）和一个或多个扩展模块（FM）组成。通常中央处理单元安装在主机架上，扩展模块安装在扩展机架上，也就是说一台 S7-300 PLC 的机架可由一个主机架和一个或多个扩展机架组成。安装模块的机架（包括主机架和扩展机架）是一种导轨，可以在该导轨上安装 S7-300 系统的所有模块。

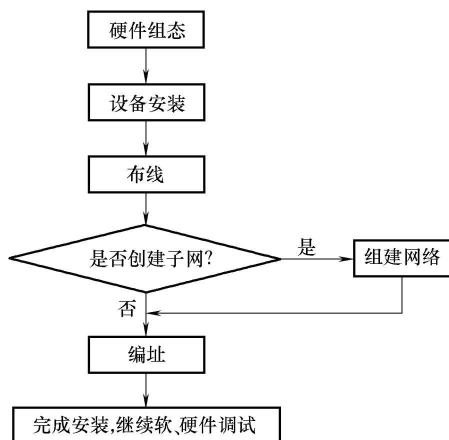


图 2-13 S7-300/400 系列 PLC 的安装步骤

S7-300 的安装方向有两种：水平安装和垂直安装。通常水平安装允许的环境温度为 0～60℃，垂直安装允许的环境温度为 0～40℃。不管安装方向如何，CPU 和电源模块必须安装在机架的左侧（水平）或底部（垂直）。

采用单机架还是多个机架进行模块安装，用户可根据实际需求来进行选择。结构紧凑、需要节约空间、只使用一个中央处理单元模块或者所需处理的信号比较少时，通常采用单机架就能完成硬件组态。

如果所需处理的信号量比较多或者机架上没有足够的插槽时，建议采用多个机架进行硬件组态。



采用单机架时应注意：

在 CPU 右边安装的模块不能超过 8 个，并且安装在单机架上全部模块的 S7 背板总线上的总电流不能超过 0.8A（安装有 CPU312、CPU312C 和 CPU312IFM）或 1.2A（除安装有 CPU312、CPU312C 和 CPU312IFM 以外）。

采用多机架时，需要使用接口模块（IM）将这些机架有机组合起来。除 CPU312、CPU312 IFM、CPU312C 和 CPU313 外，其他型号的 S7-300 系列的 CPU 都能够使用多机架。



使用多机架时应注意：

中央处理单元必须安装在机架 0 上；机架的槽 1 安装电源模块，槽 2 安装 CPU 模块，槽 3 安装接口模块；每个机架上最多只能安装 8 个信号模块，并且这些模块只能从槽 4 开始进行安装。

2.2.2　电气保护措施及接地

在一个参考电位接地的 S7-300 组态中，通过接地或接地导线的方法消除所产生的干扰电流。为此，可根据使用的 CPU，通过一根跳接线或接地滑块（CPU312IFM 或 CPU31xC 除外）来实现，如图 2-14 所示。

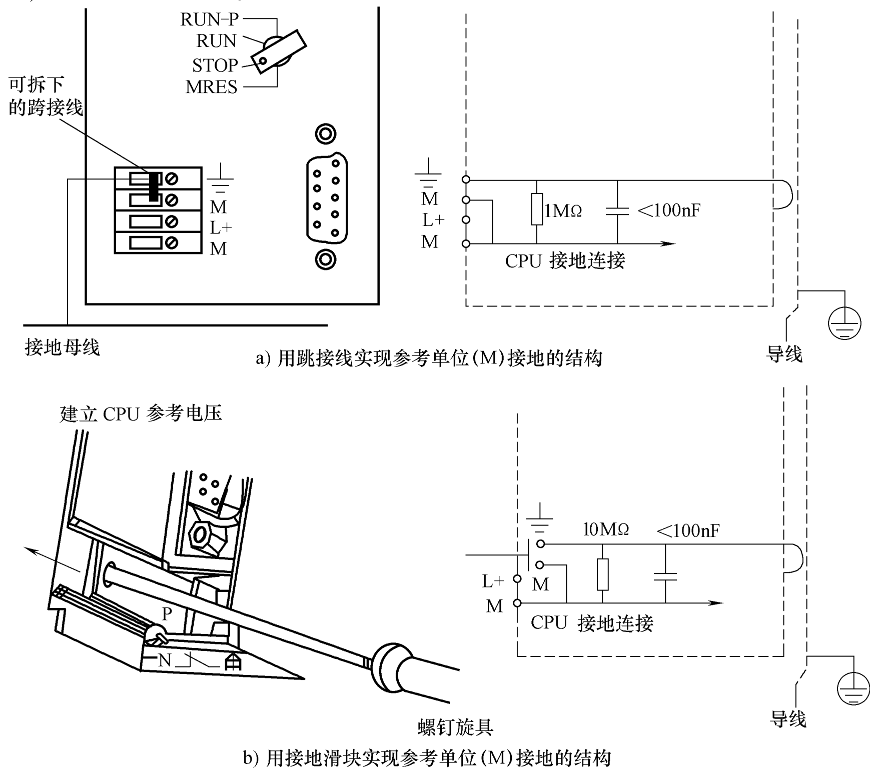


图 2-14　参考电位接地的结构

使用跳接线实现参考电位接地时，不能将跳接线从 CPU 上拆下，为了建立浮动参考电动势，需将 CPU 上的 M 端子和功能地上的跳接线去掉；如果没有安装跳接线，那么 S7-300 的参考电位是通过 RC 电路和导轨与保护性接地导体内部相连接的，这样将寄生的高频电流放电并将静电放掉。

使用接地滑块实现浮动参考电位时，可在图 2-14b 中用螺钉旋具将 CPU 上的接地滑块向前推动到位。

绝缘或非绝缘模块：在交流负载电路或有单独参考电位的直流负载电路中可使用绝缘模块。在带绝缘的模块结构里，控制回路的参考电位 M 和负载回路的参考电位是电绝缘的。在非绝缘的模块结构里，控制回路的参考电位 M 和负载回路的参考电位是非电绝缘的。

接地：低电阻接地连接可以减少短路或系统故障时的电击危险，低阻抗连接可以降低干扰对系统的影响或干扰信号的发射。因此，有效的电缆和设备屏蔽非常重要。

所有防护等级为 1 的设备以及所有大型金属部分都必须连接到保护接地，这样可确保系统用户免遭电击并且还可以消除来自外部电源线以及连接到 I/O 设备电缆的信号线的干扰。

2.3 安装

2.3.1 安装导轨

安装导轨时应留有足够的空间用于安装模块和散热器，例如模块上下至少应有 40mm 的空间，左右两侧至少应有 20mm 的空间。

在导轨和安装表面（接地金属板或设备安装板）之间会产生一个低阻抗连接。如果安装表面已涂漆或经阳极氧化处理时，应使用合适的接触剂或接触垫片以减小接触阻抗。

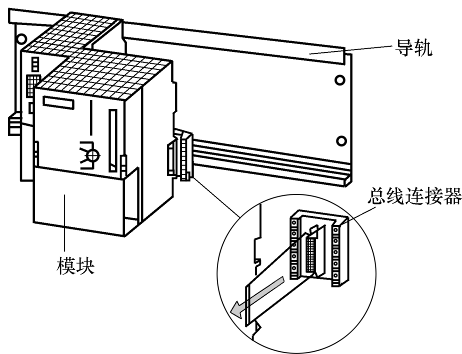


图 2-15 插入总线连接器

2.3.2 将模块安装在导轨上

固定好导轨后，在导轨上安装模块时，应从导轨的左边开始，按先安装电源模块，再安装 CPU 模块，最后安装接口模块、功能模块、通信模块、信号模块的顺序进行。安装好所有模块后，将钥匙插入 S7-300 CPU 模式选择器开关中。

1. 插入总线连接器（见图 2-15）


注意：

插入总线连接器需从 CPU 开始，最后一个模块不能安装总线连接器，应取出。

2. 安装模块

首先将模块悬挂在导轨上，如图 2-16 中的步骤 a 所示，然后将模块块划向左边，并向下安装模块，如图 2-16 中的步骤 b、c 所示。

3. 螺钉固定模块（见图 2-17）

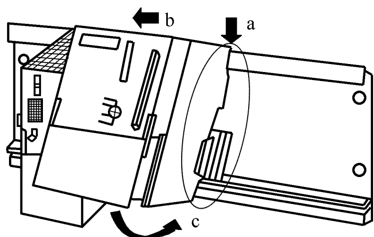


图 2-16　安装模块

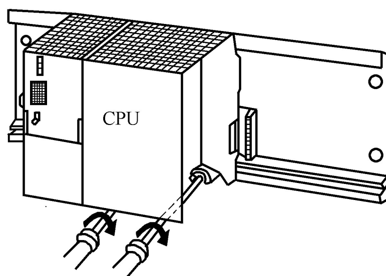


图 2-17　固定模块

2.3.3　对模块贴标签

模块安装好后，应给每一个模块指定槽号。通常第 1 槽为电源模块，第 2 槽为 CPU 模块，第 3 槽为接口模块（IM），第 4 ~ 11 槽为信号模块。槽号标签包括在 CPU 中。

对模块指定槽号的方法如图 2-18 所示，先将相应的槽号放在每个模块的前面，再将销钉插入模块的开口处（a），最后将槽号压入模块中（b），槽号从轮子上断开。

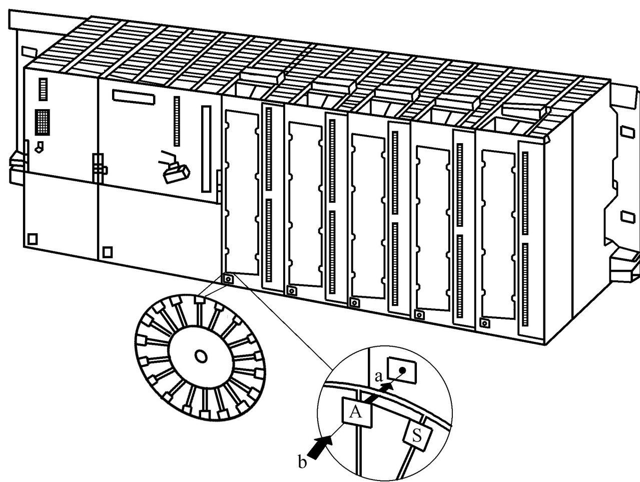


图 2-18　将槽号插入模块中

**注意：**

在不改变硬件配置的情况下，能用 SM321-1CH20 代替 SM321-1CH80 吗？

SM321-1CH20 和 SM321-1CH80 模块的技术参数是相同的。区别仅在于 SM321-1CH80 可以应用于更广泛的环境条件。因此您无需更改硬件配置。

2.4 接线

2.4.1 保护接地导线和导轨的连接

1. 保护接地导线和导轨的连接

连接保护接地导线至导轨，应使用 M6 保护导线螺栓。为保证保护接地导线的低阻抗连接，应使用尽可能短的低阻抗电缆连接到一个较大的接触表面上。保护接地导线的横截面积应大于 $10mm^2$。

2. 连接电源模块和 CPU 模块

在连接电源模块和 CPU 模块之前，应先检查一下线路电压选择器开关是否设置为所需线路电压。连接电源模块和 CPU 模块的步骤如下：

1）打开 PS307 电源模块及 CPU 模块的前盖。

2）松开 PS307 电源模块上的松紧件。

3）剥开电源线大约 11mm 长度，并连接到 PS307 的“L1”、“N”的接地导线端子。

4）旋紧松紧件。

5）对 CPU 进行接线时，分两种情况进行（见图 2-19）：对于 CPU31xC 型号的 CPU，应先剥开 CPU 电源连接导线大约 11mm 的长度，然后将 PS307 的端子“M”和“L +”连接到 CPU 的端子“M”和“L +”；对于其他型号的 CPU，插入连接梳，并拧紧即可。

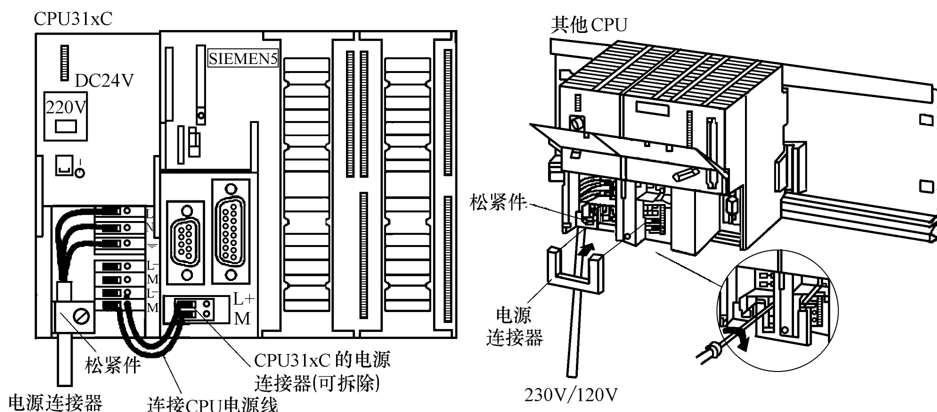


图 2-19 连接电源模块和 CPU 模块

6）盖上 PS307 电源模块及 CPU 模块的前盖板。

2.4.2 前连接器接线及插入

1. 连接

前连接器用于连接系统中的传感器和执行器到 S7-300 PLC。根据针数的不同，前连接器分为 20 针和 40 针两种类型；按连接方式的不同，又可分为弹簧负载型端子和螺钉型端子。对于 CPU31xC 和 32 通道信号模块，需要使用 40 针前连接器。

对于 20 针的前连接器，接线通常使用以下步骤：

1）将电缆的捆扎带穿入前连接器。

2）如果需要将电缆从模块的底部引出来时，从端子 20 开始，一直到端子 1 穿过前连接器；如果不需要，则从端子 1 开始，一直到端子 20 穿过前连接器。

3）拧紧未使用的端子。

4）将电缆捆扎带抽紧，从左边拉出捆扎带的尾部，使结构紧凑。

对于 40 针的前连接器，接线通常使用以下步骤：

1）如果需要将电缆从模块的底部引出来时，从端子 40 或 20 开始，从端子 39、19、38、18 等穿过接线连接器，直到到达端子 21 和 1；如果不需要，则从端子 1 或 21 开始，从端子 2、22、3、23 等穿过接线连接器，直到到达端子 20 和 40。

2）拧紧未使用的端子。

3）将电缆线穿入捆扎带。

4）将电缆捆扎带抽紧，从左边拉出捆扎带的尾部，使结构紧凑。

2. 插入

若前连接器接好线后，可按如下步骤将其插入模块。

对于 20 针前连接器，如图 2-20 所示。将开启机构推在模块上方（见图 2-20a 处）；在开户位置将前连接器插入模块（见图 2-20b 处），开户机构释放后自动返回初始位置。盖上前盖板（见图 2-20c 处）。


注意：

将前连接器插入模块中时，可以在前连接器中安装一个编码机构，以保证下次替换模块时不会产生差错。

3. 模块 I/O 标签

标签条可以记录模块 I/O 与系统传感器或执行器之间的分配情况。插入标签条的步骤：先在标签条上标记传感器或者执行器的地址，然后将标签条滑入相应模块的前面板内。

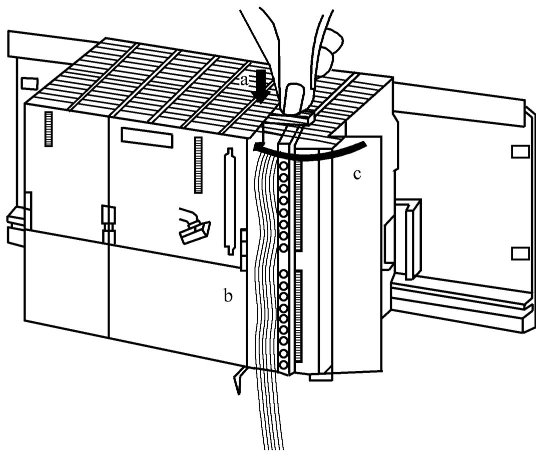


图 2-20　20 针前连接器

2.4.3　安装屏蔽连接元件及屏蔽电缆

由于屏蔽连接元件直接与导轨接触，使用屏蔽连接元件可以很容易地将 S7 模块的所有屏蔽电缆连接到地。安装屏蔽连接元件时，可按以下步骤进行：

1）将固定支架的两个螺栓推到导轨底部的滑槽里；

2）将支架固定在屏蔽电缆需端接的模块下面；

3）将固定支架旋紧到导轨上；

4）屏蔽端子下面带有一个开槽的金属片，将屏蔽端子放在支架一边，然后向下推屏蔽端子到所要求的位置。

在连接前连接器时，在屏蔽端子后面应留有足够长的电缆。这样，在断开前连接器时，用打开屏蔽连接元件。

2.5 寻址

2.5.1 模块通道寻址方式

基于槽定义的模块寻址为默认设置，也就是说 STEP 7 为每个槽号都指定有一个确定的模块起始地址。数字或模拟模块具有不同的地址，图 2-21 所示为安装在 4 个模块机架上的

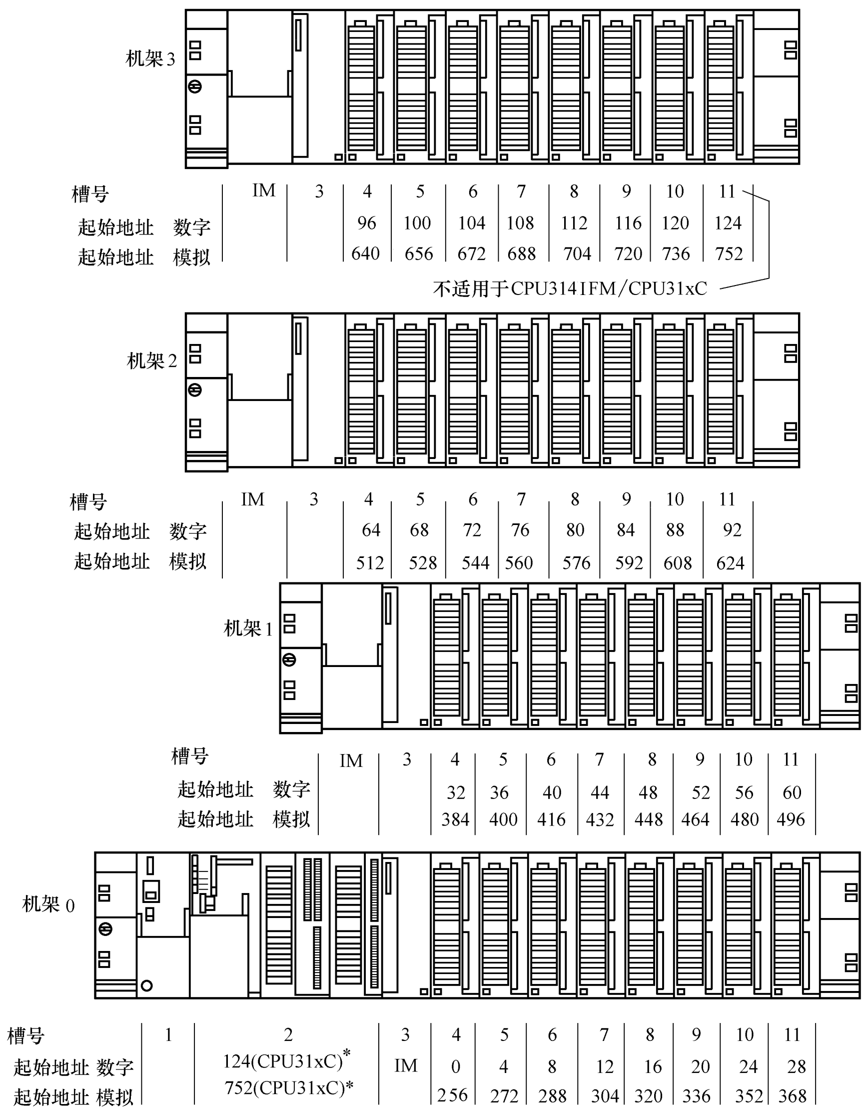


图 2-21　4 个模块机架上的 S7-300 系统

S7-300 系统以及其模块起始地址。注意，对于 CPU314IFM 和 CPU31xC，用户不能将模块插入模块机架 3 的 11 号槽中，该地址预留给集成的 I/O。

2.5.2　寻址信号模块

一个数字量模块的输入或输出地址由字节地址和位地址组成，其中字节地址取决于其模块起始地址，位地址是印在模块上的数码号。

图 2-22 所示为一块数字量模块插在 4 号槽里（模块起始地址是 0）。由于在此机架上没有接口模块，所以没有设置 3 号槽。由于第一块数字量模块插在 4 号槽，所以随后的数字量模块，其起始地址每一槽增加 4。

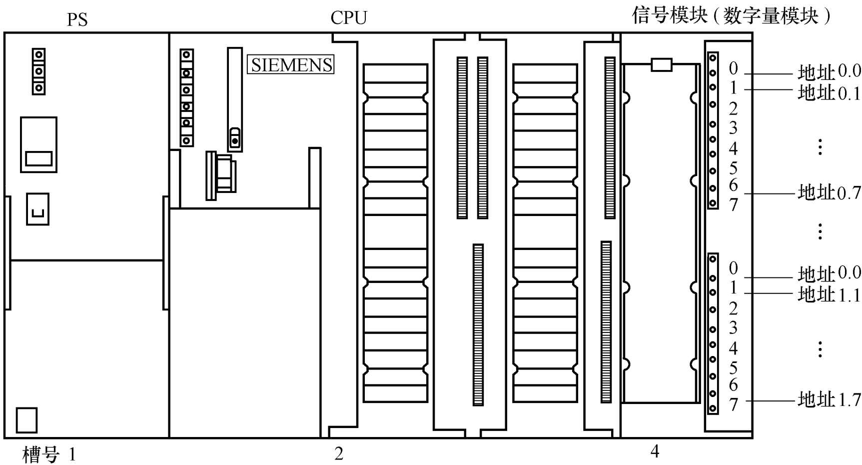


图 2-22　4 号槽中数字量模块的 I/O 地址

模拟量输入通道或输出通道的地址总是一个字地址，其中通道地址取决于模块的起始地址。图 2-23 所示为一块模拟量模块插在 4 号槽里（模块起始地址是 256）。由于在此机架上没有接口模块，所以没有设置 3 号槽。由于第一块模拟量模块插在 4 号槽，所以随后的模拟量模块，其起始地址每一槽增加 16。

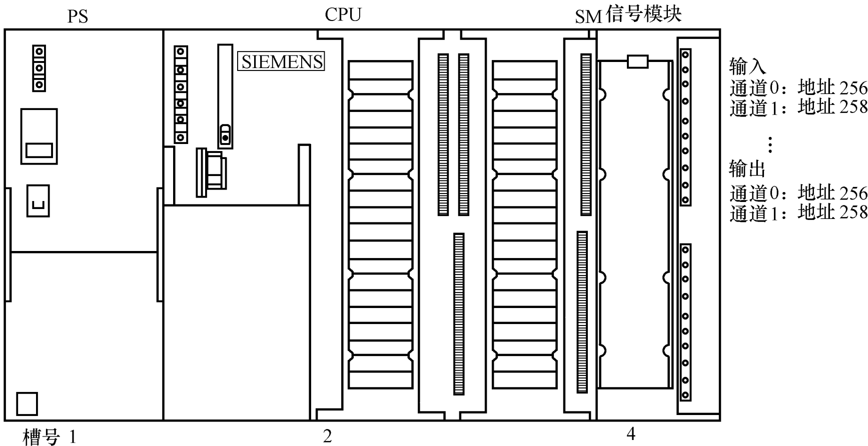


图 2-23　4 号槽中模拟量模块的 I/O 地址

2.6 仿真软件的安装

2.6.1 安装 PLCSIM

西门子 S7-PLCSIM 仿真软件是 STEP 7 的可选软件工具，安装后集成在 STEP 7 中。它能够在 PG/PC 上模拟 S7-300、S7-400 系列 CPU 中用户程序的执行过程，可以在开发阶段发现和排除错误。另外，S7-PLCSIM 可供不具备硬件设备的读者在学习时使用。

如果未安装该软件，则 SIMATIC Manager 工具栏中的（模拟）按钮处于失效状态；安装了该软件，则在 SIMATIC Manager 工具栏中的按钮处于有效状态。

使用 S7-PLCSIM 进行项目仿真时，不需任何 S7 硬件（CPU 或信号模块），使用 S7-PLCSIM 可以像对真实 PLC 硬件一样，对模块 CPU 进行程序下载、测试和故障诊断，具有方便和安全等特点，非常适合前期的工程测试。通过 S7-PLCSIM，STEP 7 的工作过程与真实的 PLC 相比差别很小。

2.6.2 PLCSIM 与真实的 PLC 的差别

S7-PLCSIM 提供了方便、强大的仿真模拟功能。与真实 PLC 相比，它灵活性更高，提供了许多 PLC 硬件无法实现的功能，使用也更方便。但是软件毕竟无法完全取代真实的硬件，不可能实现完全的仿真。用户利用 S7-PLCSIM 进行模拟调试时，必须了解它与真实 PLC 系统的差别。S7-PLCSIM 的下列功能在实际 PLC 上无法实现。

1）程序暂停/继续功能。暂停命令可以停止模拟 CPU 的运行，并且可以在暂停的指令处恢复程序执行。

2）操作模式。尽管模拟 CPU 可以像真实 CPU 一样进行操作模式选择（STOP、RUN、RUN-P），但是在模拟 CPU 的 STOP 模式下，输出的状态不发生变化。

3）立即响应。当操作对象的参数变化时，在模拟 CPU 中的存储器内容立即进行修改，而不必等到输入采样或输出刷新阶段时再进行修改。

4）程序执行周期。在模拟 CPU 中，可以选择单次扫描（一次操作只执行一个扫描周期后，等待下一次操作），或者选择连续扫描。

5）定时器操作。在模拟 CPU 中，允许定时器自动运行，允许手动输入定时值。可以对各个定时器进行单独复位或一起复位。

6）可以手动触发中断组织块。在模拟 CPU 中，可以手动触发中断组织块 OB40 ~ OB47，OB70、OB72、OB73、OB80、OB82、OB83、OB85、OB86。

7）过程映像和外部存储器。在模拟 CPU 中，当对过程输入值做出改变时，S7-PLCSIM 立即将其复制到外部存储器中。通过这种方法，在下次扫描开始，当外部输入值被写到过程映像寄存器时，所考虑的变化不会丢失。同样，当对过程输出值做出改变时，变化值会立即写入外部输出存储器。

8）诊断缓冲区。S7-PLCSIM 不支持写到诊断缓冲区中所有的错误信息。例如，不能模拟 CPU 中的电池损坏、EPROM 的错误。然而 S7-PLCSIM 可以模拟大多数的编程错误和 I/O

错误。

9）转换操作方式。当从 RUN 变为 STOP 模式时，I/O 不会进入安全状态。

10）不支持功能模板（FM）。

11）不支持对等通信。S7-PLCSIM 只模拟单机系统，不支持多 CPU 的网络通信模拟功能。

12）S7-PLCSIM 可支持 4 个累加器的模拟。

13）在 I/O 中的差别。真实的 S7-300 系列 CPU 是自动配置 I/O 的，一旦模板插入到机架中，CPU 可以自动识别。在 S7-PLCSIM 中，模拟 CPU 不能复制自动配置特性。如果要从自动配置了 I/O 的 S7-300 CPU 中下载程序到 S7-PLCSIM，系统数据不包括 I/O 配置。如果在 S7-300 的程序中使用 S7-PLCSIM，为了使 CPU 识别可支持 I/O 模板，首先要下载硬件配置。要下载硬件配置到 S7-PLCSIM，首先要创建一个项目，然后将硬件配置复制到这个项目中，再下载硬件配置到 S7-PLCSIM 中，之后才能下载程序块到 S7-PLCSIM。

第 3 章

S7-300/400快速入门

3.1　STEP 7 软件入门

在使用 STEP 7 之前，为了适合自己的习惯，可以对 STEP 7 软件的使用环境进行设置。双击 Windows 桌面上的“SIMATIC Manager”图标，如图 3-1 所示。

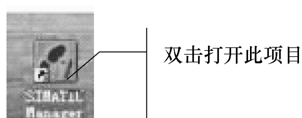


图 3-1　从桌面“SIMATIC Manager”图标打开 STEP 7 管理界面

打开 STEP 7 管理界面，默认自动启动向导如图 3-2 所示。



图 3-2　SIMATIC　管理器创建项目向导界面

在图 3-2 中单击“完成”，直接以预览框显示的项目生成，如图 3-3 所示，若单击“取消”，直接进入最近一次打开的项目画面。

在 SIMATIC 管理器项目管理界面中，单击菜单栏的“选项”→“自定义”，如图 3-3 所示。



图 3-3　SIMATIC 管理器项目管理界面

在自定义设置界面单击“常规”选项卡，如图 3-4 所示。

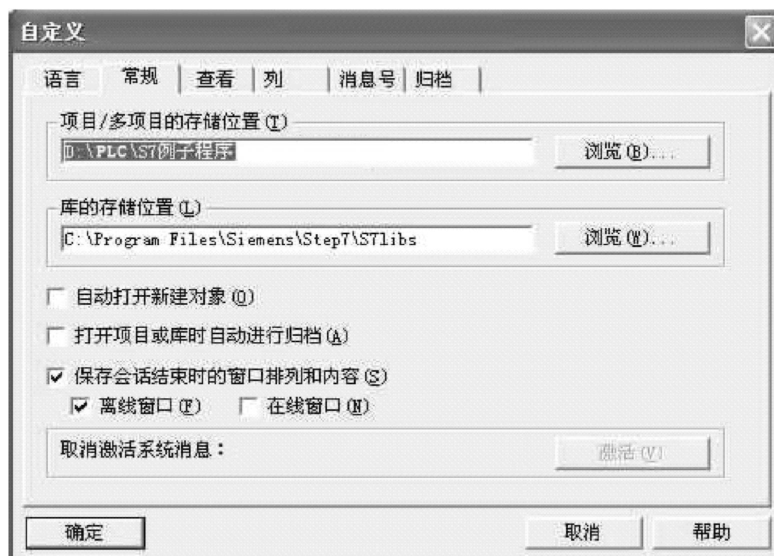


图 3-4　常规选项卡

单击“语言”选项卡，如图 3-5 所示，设置适合自己要求的每一项，然后单击“确定”。



图 3-5 语言选项卡

3.2 STEP 7 硬件组态

3.2.1 创建一个项目

1）双击 Windows 桌面上的 SIMATIC 管理器图标打开 STEP 7 软件，默认自动启动向导，如图 3-6 所示。单击“预览”，可以选择显示或隐藏正在创建的项目结构的视图。



图 3-6 SIMATIC 管理器向导（选择创建项目的途径）

2）如要转到下一个对话框，单击“下一步”，出现如图 3-7 所示的向导画面。

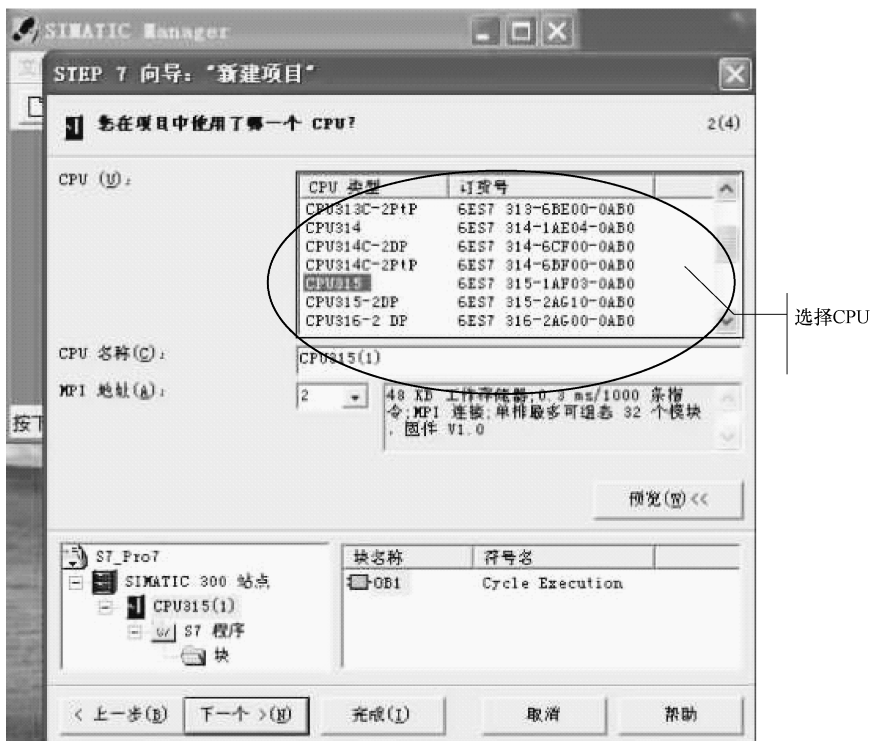


图 3-7　SIMATIC 管理器向导

**注意：**

如果向导没有自动启动，请选择菜单命令“文件”→“新建项目”向导。

如不要转到下一个对话框，可以单击“完成”，自动创建预览框所示的项目。如不要转到下一个对话框，也可以单击“取消”，根据需要自行创建项目的每个目录。

对于“S7_Pro7”项目，选择 CPU 315，原则上可以选择 CPU 类型表中任意一款 CPU。MPI 地址的默认设置为 2，如果需要可以选择 MPI 地址为 2 以上的一个值。单击“下一个”确认设置，进入下一个向导设置画面，如图 3-8 所示。

选择 MPI 地址是为了使 CPU 与编程设备或 PC 之间进行通信，所以必须要设置 MPI 地址（多点接口）。在图 3-8 向导中，选择组织块 OB1（如果尚未选中），OB1 代表最高的编程层次，它负责组织 S7 程序中的其他块，一个程序中必须要有 OB1。选择合适的一种编程语言：指令表（STL）、梯形图（LAD）或功能块图（FBD）。

3）单击“下一个”确认设置，进入下一个向导设置画面，如图 3-9 所示。

4）在图 3-9 向导中，在“项目名称”域中双击选中默认的名称，更改项目的名称为“PLC 第一个项目”或使用默认名称。最后单击“完成”，将自动按照向导的各个目录生成项目，同时自动打开 SIMATIC 管理器和刚刚创建的“PLC 第一个项目”的窗口，如图 3-10 所示。



图 3-8 SIMATIC 管理器向导（选择程序块）

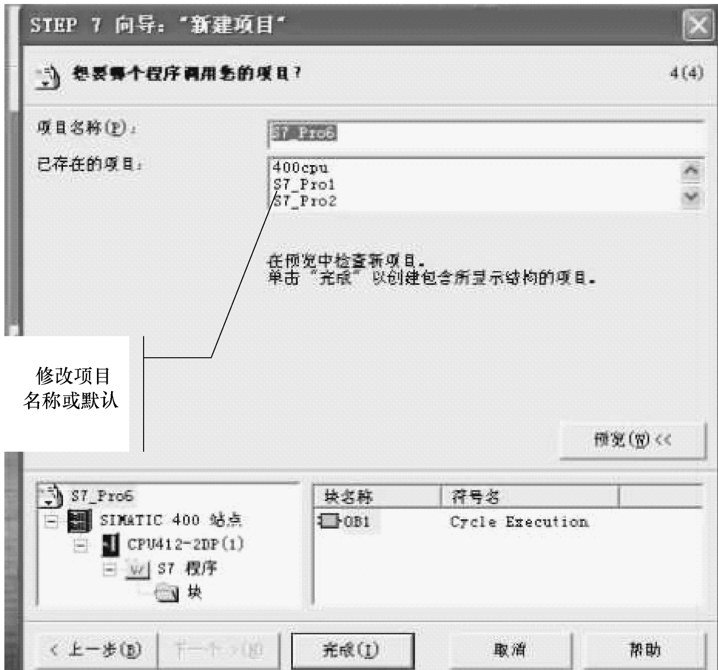


图 3-9 SIMATIC 管理器向导

3.2.2 硬件组态

硬件组态的任务是在 STEP 7 的配置机架（HW Config）画面中，组态一个与实际硬件相



图 3-10　完成后的向导配置

同的硬件系统，使得软件与硬件一一对应。机架上所有模块的参数在组态过程中使用软件设置，CPU 参数保存在系统数据（SDB）中，其他模块的参数保存在 CPU 中。

在设计一个控制系统之前，按照控制系统性质决定使用硬件及网络配置，然后在硬件组态中，定义每个模块的参数，包括 I/O 地址、网络地址及通信波特率等参数。

S7-300 实际硬件中机架配置模块之间必须无间隙地插入到机架中。例外：对于只有一个机架的安装，软件组态表里槽 3 保持为空（为接口模块保留），槽 4 ~ 11 可以组态其他的模块。但实际的硬件中将不允许有间隙，如果实际硬件有间隙，背板总线将被中断，如图 3-11 所示。

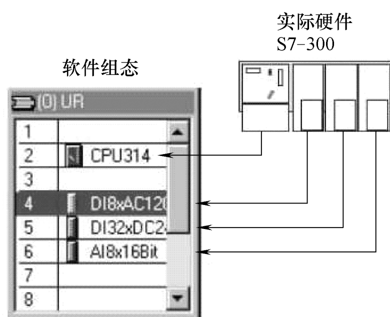


图 3-11　软件组态与实际硬件对应视图

3.2.3　配置机架

1. 配置中央机架

我们以 S7-300 为例说明中央机架的组态过程及步骤。

1）在新建立的项目中，如图 3-10 所示，单击“SIMATIC 300 站点”，右边自动弹出硬件“ ”和 CPU“ ”的图标，双击“ ”硬件符号，将自动弹出配置机架（HW Config）的画面，如图 3-12 所示。

2）在图 3-12 所示的配置中央机架画面中，将创建项目时所选择的 CPU 显示出来，对于“PLC 第一个项目”是 CPU 315。现在准备组态电源及所需要的模块，本例选用 PS 307 2A 电源、SM321 DI32xDC24V 输入模块及 SM322 DO32xDC24V/0.5A 输出模块。

3）在图 3-12 所示的配置中央机架组态画面中，双击硬件目录中“SIMATIC 300”，打开 S7-300 的硬件目录，如图 3-13 所示。

4）在图 3-13 所示 S7-300 的硬件目录画面中，双击“PS-300”电源目录，找到 PS 307



图 3-12　配置中央机架

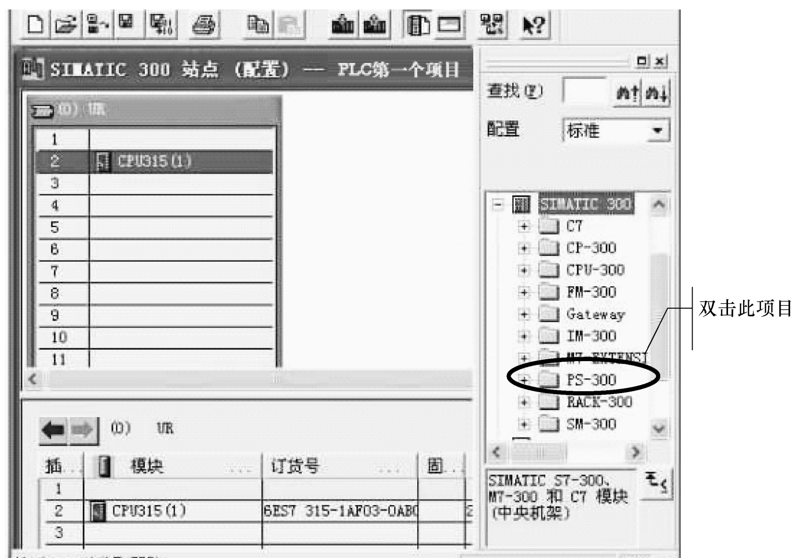


图 3-13　S7-300 的硬件目录

2A，然后将该模块拖放到插槽 1 的位置，如图 3-14 所示。

5）在图 3-13 所示的 S7-300 的硬件目录画面中，双击“SM-300”打开 I/O 模块目录，



图 3-14　组态电源模块

在 I/O 模块目录中双击“DI-300”打开数字输入模块目录，找到 SM 321 DI32xDC24V，然后将该模块拖放到插槽 4 的位置，如图 3-15 所示。



图 3-15　组态数字输入模块

6）在图 3-13 所示的 S7-300 的硬件目录画面中，双击“SM-300”打开 I/O 模块目录，在 I/O 模块目录中双击“DO-300”打开数字输出模块目录，找到 SM 322 DO32xDC24V/0.5A，然后将该模块拖放到插槽 5 的位置，如图 3-16 所示。



图 3-16　组态数字输出模块

如果需要在一个项目中修改模块的参数（例如地址），双击机架插槽相应的模块，在弹出的对话框中更改，但是应该在确信知道改变这些参数会对可编程序控制器有何影响时方可改变它们，一般情况下使用默认值即可。

最后在图 3-16 所示画面中，使用菜单命令保存和编译，或单击“”图标进行保存和编译，向 CPU 传送准备好的数据。经过保存和编译操作后，在管理画面的“块”文件夹中将会出现系统数据“系统数据”的符号。

2. 配置扩展机架

S7-300 CPU 最多可以安装包括中央机架在内 4 个机架，最多可以安装 32 个模块（除电源、CPU 及接口模块以外）。S7-300 CPU 在配置扩展槽时各槽可以插入的模块如表 3-1 所示。

表 3-1　各槽号可以插入的模块

机架	槽 1	槽 2	槽 3	槽 4～11
机架 0（中央机架）	电源或为空	CPU	接口模块	信号、功能模块、通信处理器或为空
机架 1～3		为空	接口模块	

1）占位模块（DM 370）：占位模块是代替随后要使用的模块插入到插槽中而暂时插入，起到占用组态地址分配的作用。根据开关设置，这种模块既可以为模块保留地址空间，也可以不保留。例如，为数字量输入/输出模块保留地址空间，而对接口模块则不保留。

2）数字仿真模块（SIM 374 IN/OUT 16）：数字仿真模块可用于仿真数字量输入和输出。在“硬件目录”窗口中找不到该模块，必须将想要仿真的模块放入组态表，而不是放入 SIM 374 中，如表 3-2 所示。

表 3-2　数字仿真模块

SIM 374 IN/OUT 16 的开关设置	要放入的模块
16xOutput	6ES7322-1BH00-0AA0
8xOutputx8Input	6ES7323-1BH00-0AA0
16xInput	6ES7321-1BH00-0AA0

3）组态分布式 I/O。分布式 I/O 指主站系统，其包含通过总线电缆相连、通过 PROFIBUS-DP 协议互相通信的 DP（分布式 I/O）主站和 DP 从站。

分布式 I/O 系统中的插槽编号及模块如表 3-3 所示。

表 3-3　分布式 I/O 系统中的插槽编号及模块

<table>
<tr><th>机架</th><th>槽 1</th><th>槽 2</th><th>槽 3</th><th>槽 4 ~ 11</th></tr>
<tr><td>机架 0(中央机架)</td><td rowspan="2">电源或为空</td><td>CPU</td><td>接口模块</td><td rowspan="2">I/O(输入/输出)</td></tr>
<tr><td>分布式 I/O 设备</td><td>DP 接口
face 模块</td><td>接口模块</td></tr>
</table>

分布式 I/O 具体的组态过程请参看本书第 7 章中的相关内容。

4）在硬件组态画面更换模块。可以在硬件组态画面（HW Config）来修改站组态。例如，希望为一个站更换具有新订货号的模块，处理步骤如下：

① 使用拖放操作从硬件目录窗口将模块拖到已放置好的旧模块上。

② 从实际机架上相应位置安装新模块，新模块会立即采用已插入的模块的参数。



注意：什么是自由分配 I/O 地址？

地址的自由分配意味着用户可对每种模块（SM/FM/CP）自由地分配一个地址。地址分配在 STEP7 里进行。先定义起始地址，该模块的其他地址以它为基准。

自由分配地址的优点：因为模块之间没有地址间隙，就可以优化地使用可用地址空间。在创建标准软件时，分配地址过程中可以不考虑所涉及的 S7-300 的组态。

3. 2. 4　硬件更新

西门子的硬件更新比较快，每一个 STEP 7 版本都不可能包含将来的硬件。从 STEP 7 V5. 2 开始，STEP 7 提供了硬件更新的功能。

当在组态硬件时，在硬件目录里无法找到需要配置的硬件（一般以订货号为准）时，说明需要更新硬件目录了。硬件更新一般从 Internet（互联网）上更新比较方便，方法如下：

首先打开 STEP 7 的硬件组态界面，单击菜单栏的“选项”→“安装 HW 更新”，如图 3-17所示。

将自动弹出要求选择更新的途径，如图 3-18 所示。



图 3-17 打开硬件更新的路径



图 3-18 更新途径对话框

选择“从 Internet 下载”，然后单击“执行”，自动进入搜索需要下载的硬件，如图 3-19 所示。



图 3-19　下载过程中

搜索完毕后，自动弹出要求选择下载的硬件列表，选择需要下载的硬件，如图 3-20 所示，然后单击“下载”，下载完毕后自动弹出已经下载的硬件列表，如图 3-21 所示。选择需要安装的硬件并单击“安装”，按照提示进行安装。安装完毕重新打开 STEP 7，新安装的硬件可以组态使用了。



图 3-20　选择需要下载的硬件



图 3-21　选择需要安装的硬件

3.3　在线调试

3.3.1　设置 PG/PC 接口

作为在线调试的第一步，必须在编程设备（Programming Device，包括编辑器或编程计算机，以下通称 PG）与实际 PLC 的 CPU 之间建立硬件连接，为在线调试提供数据交换与通信的线路。

PG 与 PLC 的连接应使用 PLC-CPU 上的集成 MPI 或安装在 PLC 上的 MPI 模块进行。PG 与 PLC 间的连接可以采用一台 PG 与一台 PLC 的 1:1 连接形式，也可以通过 PLC 网络进行一台 PG 与多台 PLC 的 1:N 连接形式。当 1:1 连接时，一般可以直接使用 PLC 的串行通信口；当采用 1:N 连接时应使用 PROFIBUS-DP 总线。两种连接方式的具体要求如下。

1. 1:1 连接

PG 连接单台 PLC 是一种常规连接形式，其连接非常简单，PG 与 PLC 间只需要利用 S7 专用的编程电缆，将 PG 直接连接到 PLC-CPU 上的集成 MPI 或安装在 PLC 上的 MPI 模块即可，如图 3-22 所示。

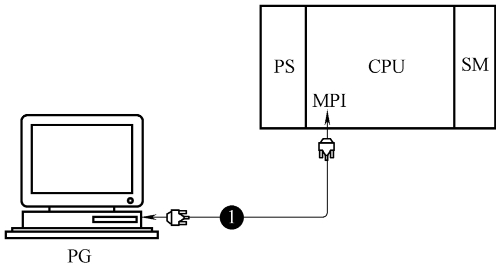


图 3-22　PG 与 PLC 的 1:1 连接

2. 1∶N 连接

在 PLC 网络系统中，PG 与 PLC 之间可以通过 PROFIBUS-DP 总线进行连接。连接可以采用如下 3 种形式。

1）将 PG 作为 PLC 网络系统的站点连接到 PLC 网络中，连接如图 3-23 所示。

2）将 PG 作为临时调试设备，在需要时与 PLC 网络进行现场连接，连接如图 3-24 所示。

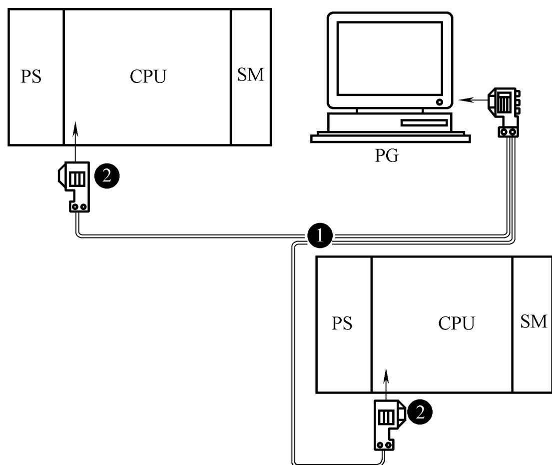


图 3-23　PG 作为站点连接到 PLC 系统
1—PRORFIBUS 总线　2—终端连接器

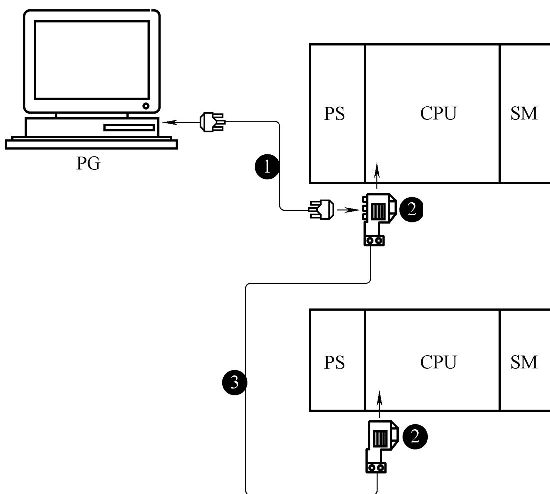


图 3-24　PG 作为调试设备的连接
1—电缆　2—总线连接器　3—PROFIBUS 总线

3）将 PG 作为远程设备，通过调制解调器（Modem）、通 信 服 务 器（Tele Service Adapter）进行远程网络连接，连接如图 3-25 所示。

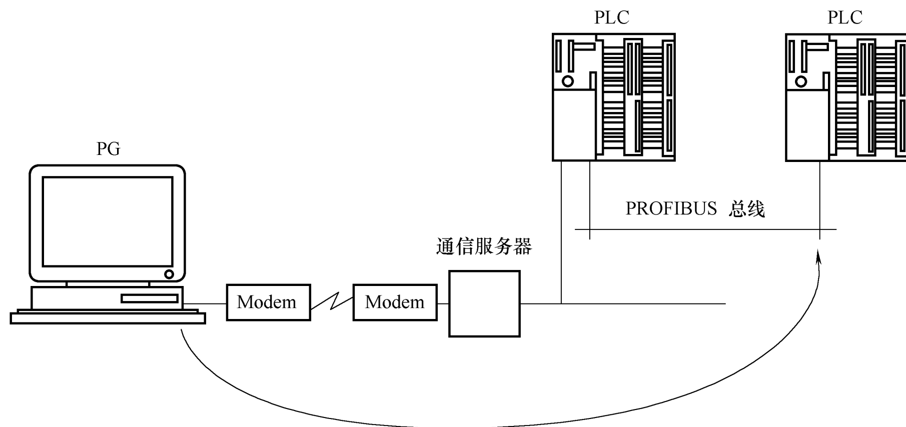


图 3-25　PG 作为远程设备的连接

在进行 PG 与 PLC 之间连接时应注意如下两点：

① MPI 地址：如 PG 不作为网络中的固定站点，而是作为临时性的服务设备使用时，推荐将 PG 以未知（unknown）节点的形式连接到 MPI 子网中，并且在 PG 上设定 MPI 地址为 0，最大 MPI 地址为 126。

② RS-485 适配器：PG 有接地与不接地两种类型。当 PG 不接地时，如果连接到同样不接地的 MPI 网络中，两者可以进行直接连接。但当 PG 接地时，PG 与 PLC 之间必须使用 RS-485 适配器进行的连接，如图 3-26 所示。

RS-485 适配器的外形如图 3-27 所示，应将 PG 连接到 RS-485 适配器的信号接地接口 1 的 A1/B1 端或 PG/OP 接口上，将不接地的 PLC-CPU 连接到 RS-485 适配器信号不接地接口 2 的 A2/B2 端。PG/OP 连接端采用的是 9 芯连接器，其信号布置见表 3-4。

表 3-4　PG/OP 连接端信号布置

插脚号	名称	作用	插脚号	名称	作用
1	—	—	5	P5V2	+5V 电源
2	M24V	24V 接地端	7	P24V	+24V 电源
3	RxD/TxD-P	数据线 B	8	RxD/TxD-N	数据线 A
4	M5V2	参考电平(0V)	9	—	—

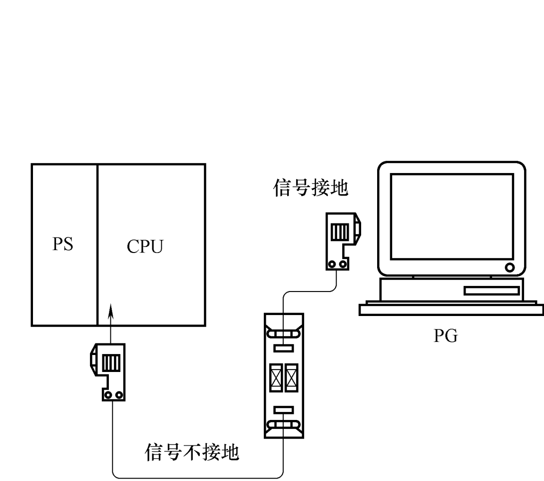


图 3-26　使用 RS-485 适配器的连接

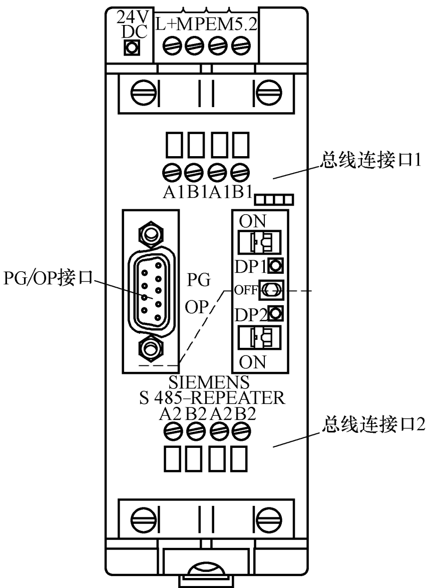


图 3-27　RS-485 适配器

3. 3. 2　建立在线连接

打开 STEP7 的项目管理画面，如图 3-28 所示，在项目管理画面里单击菜单栏的“选项”→“设置 PG/PC 接口”，进入 PG/PC 接口设置画面。

在 PG/PC 接口访问路径设置画面中，选择“PC Adapter（MPI）”，如图 3-29 所示，然后单击“属性”，进入 PC Adapter（MPI）属性设置画面。

在 PC Adapter（MPI）属性设置画面中，单击“本地连接”，设置 PG/PC 的 COM 口及传输波特率，注意这里传输波特率需要与 MPI/DP 适配器的开关位置指示的波特率一致。本例选择 COM1 及 38400（对应 MPI/DP 适配器的开关位置的 38.4），如图 3-30 所示。

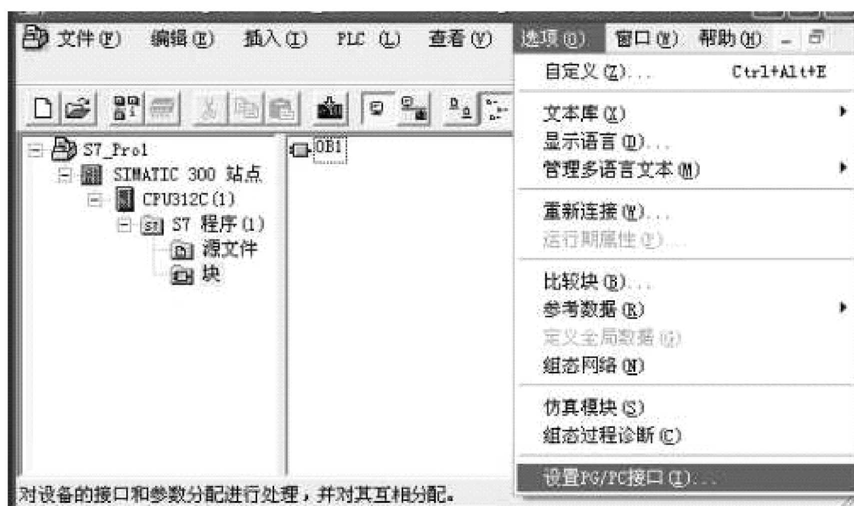


图 3-28　STEP 7 的项目管理画面



图 3-29　PG/PC 接口选择画面



图 3-30　PG/PC 接口

在 PC Adapter（MPI）属性设置画面中，单击“MPI”，设置与 CPU 中 MPI 口通信的波特率，一般选择“187.5kbps”，如图 3-31 所示。

当使用 MPI/DP 适配器把 PG/PC 的 RS232 COM1 与 CPU 的 MPI 口连接上并设置好 PC Adapter（MPI）属性参数后，就可以建立 STEP 7 与 CPU 的在线连接。



图 3-31　设置 MPI 网络参数

1. 设置离线/在线

STEP 7 中分项目管理画面、编程画面、数据查看画面和组态画面。这些画面分离线和在线两种画面。离线画面显示的是 PG/PC

存储的项目内容；在线画面显示的是 CPU 中的实际内容。

在 STEP 7 中相应画面单击“ ”图标即可显示相应的在线内容，图 3-32 所示是在线管理界面块目录的内容。

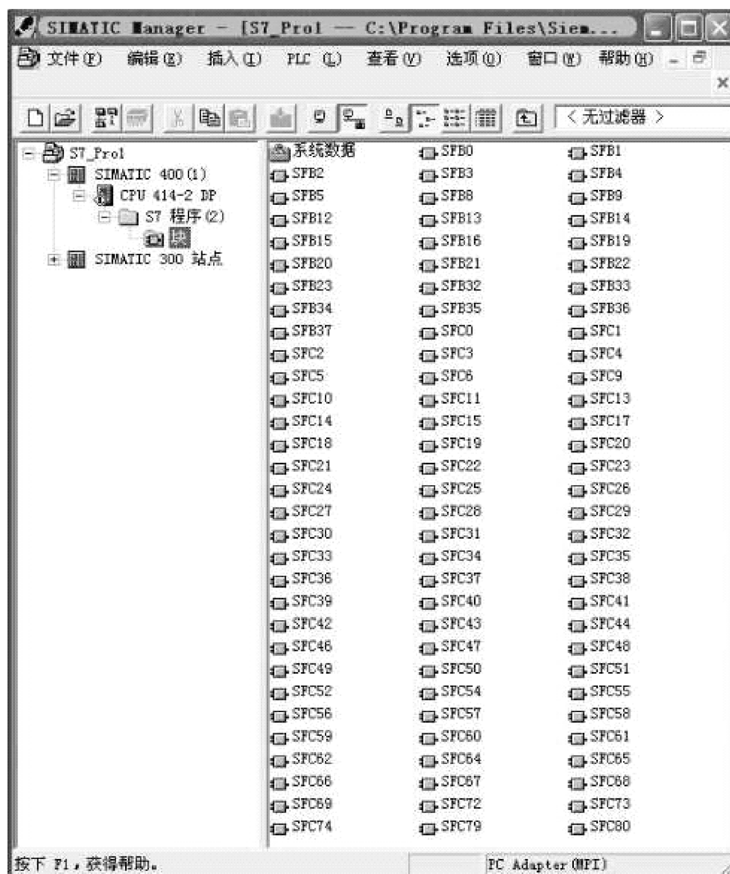


图 3-32 在线块中的目录

在 STEP 7 中相应画面单击“ ”图标即可显示相应的离线内容，图 3-33 所示是离



图 3-33 离线块中的目录

线管理界面块目录的内容。

对在线更改或删除内容，是直接修改 CPU 中的实际内容；对离线更改或删除内容，是只对当前编程设备存储内容的更改或删除。

2. 使用在线/离线进行块比较

可以对在线和离线块内容进行比较，常用这种功能进行工程项目的初步诊断，查看 CPU 的程序与保存的原程序有没有改动的地方。

首先使用编程电缆把 STEP 7 与 CPU 连接，使用 STEP 7 打开保存的原程序的块目录，然后在 STEP 7 管理界面单击菜单栏的“选项”→“比较块”，如图 3-34 所示。



图 3-34　要求在线/离线进行块比较

要求进行在线和离线块的比较，将自动弹出要求选择在线/离线块比较的类型对话框，如图 3-35 所示。

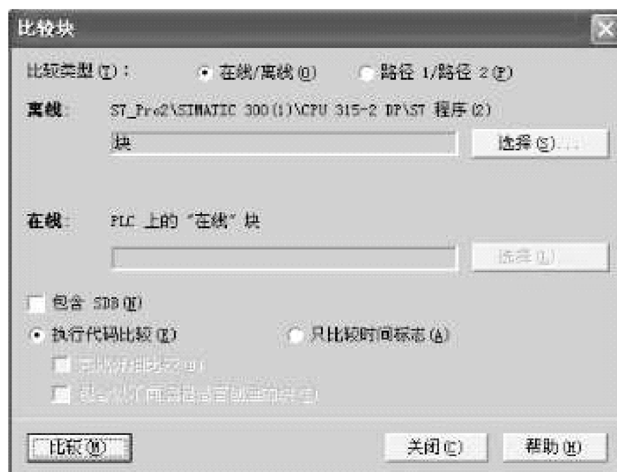


图 3-35　选择在线/离线块比较的类型

选择“在线/离线”和“执行代码比较”等项，然后单击“比较”，完成比较后自动弹出比较结果信息，如图 3-36 所示。

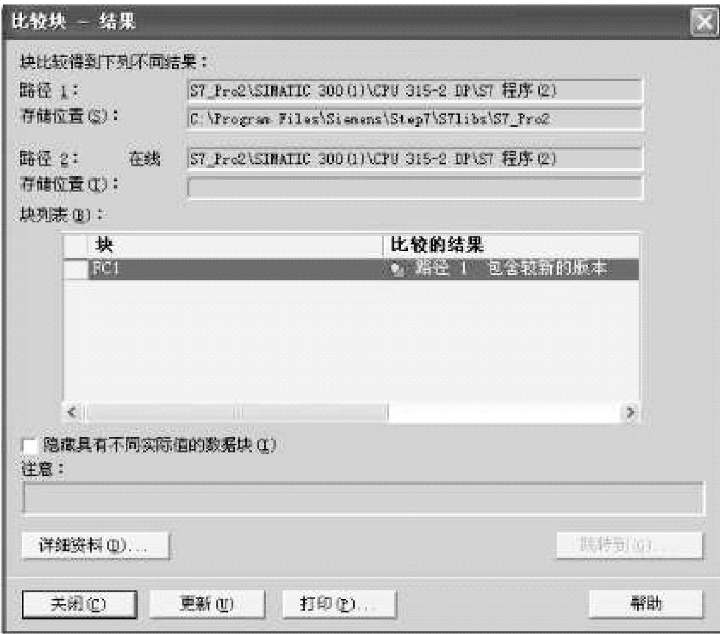


图 3-36 在线/离线块比较的结果信息

如果需要了解更详细的比较信息可以单击图 3-36 中的“详细资料”，将弹出比较结果的详细信息，如图 3-37 所示。



图 3-37 在线/离线块比较的详细信息

3.4 硬件调试与诊断

3.4.1 硬件状态指示灯

通过出现的诊断符号，如图 3-38 所示，可以浏览是否有可供模块使用的诊断消息。诊断符号说明了相应模块的状态，对于 CPU 模块也陈述了其工作模式。

当调用功能“诊断硬件”后，诊断符号将会显示在在线视图、快速视图（默认设置）

或诊断视图的项目窗口中。双击快速视图或诊断视图中的诊断符号，可启动“模块信息”应用程序来显示详细的诊断信息。这些操作关系如图 3-39 所示。

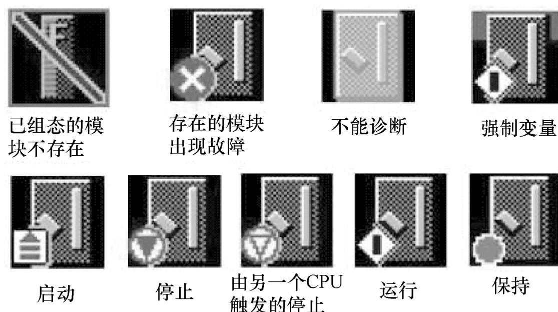


图 3-38　诊断符号

使用硬件诊断来定位故障的步骤和方法（图 3-38 硬件诊断示意图的解释）：

1）在 STEP 7 管理画面，单击菜单栏“查看”→“在线”打开项目的在线界面。

2）在 STEP 7 管理画面打开所有的站，使在其中组态的可编程模块均为可见。

3）在 STEP 7 管理画面中，视图中的 CPU 正在显示诊断符号，其指示了状态或故障。

4）在 STEP 7 管理画面中，选择要检查的站（单击表示选中）。

5）在 STEP 7 管理画面中，单击菜单栏“PLC”→“诊断/设置”→“模块信息”显示该站中 CPU 的模块信息。

6）在 STEP 7 管理画面中，单击菜单栏“PLC”→“诊断/设置”→“诊断硬件”显示该站中 CPU 和故障模块的“快速视图”。快速视图的显示已设置为默认值（单击菜单栏的“选项”→“自定义”→“查看”标签可以更改快速视图的显示设置）。

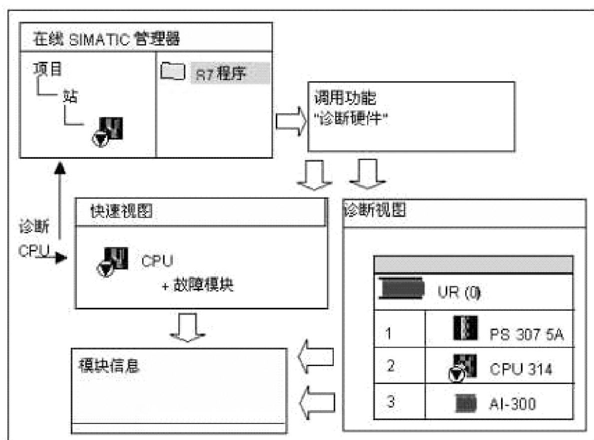


图 3-39　硬件诊断示意图



注意：错误 OB 的用途是什么？

如果发生一个所描述的错误，则将调用并处理相应 OB。如果没有加载该 OB，则 CPU 进入 STOP（例如，OB70、72、73 和 81）

S7-CPU 可以识别两类错误：

1）同步错误：这些错误在处理特定操作的过程中被触发，并且可以归因于用户程序的特定部分。

2）异步错误：这些错误不能直接归因于运行中的程序。这些错误包括优先级类的错误，自动化系统中的错误（故障模块）或者冗余的错误。

7）在快速视图中，选择需要查看的模块，然后单击“模块信息”查看该模块的信息。

8）在快速视图中，单击“在线打开站”，弹出的诊断视图包括了按照其插槽顺序排列在站中的所有模块。

9）在诊断视图中，双击机架上需要查看的模块，显示该模块信息。采用该方式，也可

获得那些没有故障因而没有显示在快速视图中的模块的信息。

在“模块信息”对话框内的各种标签中查找每个模块的信息功能。在激活状态下显示时，只显示与选中模块有关的信息。模块信息如表 3-5 所示。

表 3-5 模块信息

功能/标签	信 息	使 用 方 法
常规	显示所选择模块的标识数据,例如订货号、版本号、状态、机架中的插槽	把来自插入模块的在线信息与已组态模块的数据进行比较
诊断缓冲区	诊断缓冲区中的事件总览以及选中事件的详细信息	查找引起 CPU 进入 STOP 模式的原因,并在选中的模块上评估导致该原因的事件,通过诊断缓冲区,可以在以后分析系统中的错误,查找引起 STOP 的原因或追踪并归类单个诊断事件的发生
诊断中断	选中模块的诊断数据	评估模块故障的原因
DP 从站诊断	选中 DP 从站的诊断数据(符合 EN 50170)	在 DP 从站中评估故障原因
存储器	存储器容量,选中 CPU 或 M7 功能块的工作存储器、装入存储器和保持性存储器的当前用途	在将新模块或扩展模块传送到 CPU 之前,检查 CPU/功能模块中是否有足够可用的装入存储器,否则就压缩存储器内容
扫描周期	选中 CPU 或 M7 功能模块的最长、最短以及最后一次扫描周期的持续时间	持续检查已组态的最小周期、最大周期和当前周期
时间系统	当前时间、工作小时和同步时钟的信息(同步时间间隔)	显示和设置模块的时间与日期,并检查时间同步
性能数据	选中模块(CPU/FM)的地址区和可用的块	在创建用户程序之前以及期间,检查 CPU 是否满足执行用户程序的要求,例如,装入存储器大小或过程映像大小
“性能数据”中的块	显示选定模块供应范围内的所有可用的块类型,列出可用于该模块的 OB、SFB 和 SFC	检查用户程序可包含或调用哪些可在选中 CPU 上运行的标准块
通信	传输率、通信连接总览、通信负载以及选中模块通信总线上的最大消息帧的大小	确定可使用多少个 CPU、哪个 CPU 或 M7 F7 连接以及正在使用的数目
栈	堆栈标签:只能在 STOP 模式或 HOLD 模式中调用 显示用于选中模块的 B 栈,然后还可显示 I 栈、L 栈以及嵌套栈,并跳转到中断块中的错误位置	确定转换到 STOP 模式的原因并更正程序块(查找由于用户程序引起 STOP 的原因)

3.4.2 设置块测试环境

需要记录程序状态，可以通过设置调用环境来指定触发条件。只有在满足触发条件设置时，才会记录程序状态。设置调用环境操作如下：

1）在程序块界面单击菜单栏“调试”→“操作”，如图 3-40 所示。

在操作设置界面选择“测试操作”工作模式，如图 3-41 所示。

2）已选择“测试操作”模式后，单击菜单栏“调试”→“调用环境”，如图 3-42 所示。



图 3-40　打开"操作"模式设置界面

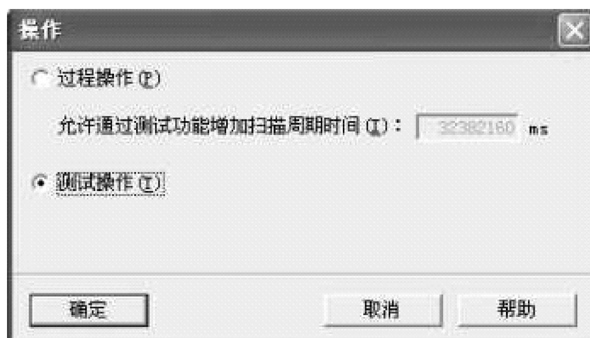


图 3-41　选择"操作"模式界面



图 3-42　打开"调用环境"界面

3）在调用环境界面中，设置触发条件，然后单击“确定”。调用环境界面相关选项的说明如表 3-6 所示。

表 3-6 调用环境界面相关选项的说明

选项	意　义
调用路径	此处可以指定调用路径，在此路径下测试的块必须被调用，以便触发状态记录
打开数据块	此处通过命名一个或两个数据块来指定调用环境，如果被测试的块用指定的数据块调用，状态就会被记录

为块调用直接指定调用环境，以使块的程序状态显示出来。操作步骤如下：

1）在程序界面单击菜单栏的“调试”→“操作”，在弹出界面中选择“测试操作”模式。

2）在线打开调用块，或在打开块后，将其下载到 PLC，并将光标放在要用的调用指令上（STL 中的 CALL 行或 LAD/FBD 中块的逻辑框）。

3）单击鼠标右键，在弹出的界面中单击“被调用的块”→“通过调用路径监视”，如图 3-43 所示。可以看到调用的块被打开，同时激活了这个背景块的状态，可以人为地满足调用条件进行程序测试。



图 3-43 直接在程序界面指定块调用环境

3.5 控制和监视变量

3.5.1 变量表

变量表具有能够存储各种不同测试情况的作用，从而在操作期间和保养维护时很容易进行测试和监控。

1）使用变量表进行监视和修改的操作步骤。创建一个新的变量表或打开一个已经存在的变量表。在块目录里把鼠标放在右边空白处，然后单击右键，在弹出的界面中单击“插入新对象”→“变量表”即可创建一个新的表，如图 3-44 所示。需要打开变量表可以双击块

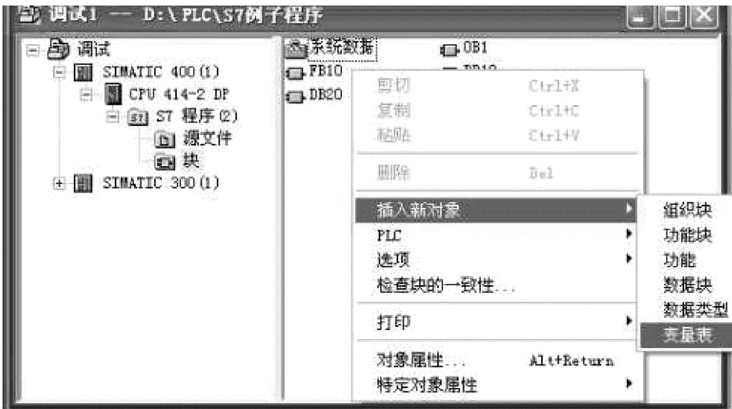


图 3-44　新建（插入）变量表路径

目录里的“ ”变量表图标，即可打开。

2）编辑或检查变量表的内容，如图 3-45 所示（如果刚新建的变量表是空白的，需要将变量内容填入）。

	地址	符号	显示格式	状态值	修改数值
1	//油泵控制　S7-400				
2	M 0.0	"油启动"	BOOL		
3	M 0.1	"油停止"	BOOL		
4	M 0.2	"油星形"	BOOL		
5	M 0.3	"油主"	BOOL		
6	M 0.4	"油三角"	BOOL		
7	M 2.0		BOOL		
8	//水泵控制 S7-400				
9	M 1.0	"水启动"	BOOL		
10	M 1.1	"水停止"	BOOL		
11	M 1.2	"水星形"	BOOL		
12	M 1.3	"水主"	BOOL		
13	M 1.4	"水三角"	BOOL		
14	//s7-300的启动/停止控制				
15	M 0.0	"油启动"	BOOL		
16	M 0.1	"油停止"	BOOL		
17	M 0.2	"油星形"	BOOL		
18					

图 3-45　编辑变量表

3）打开变量表界面，单击菜单栏的“PLC”→“连接到”，使当前变量表和所需的 CPU 之间建立在线连接，选择需要连接的 CPU，如图 3-46 所示。

“组态的 CPU”：当前已经建立的一个与 CPU 之间的连接或最近一次的连接。

“直接 CPU”：变量表所在块的 CPU 建立连接。

“可访问的 CPU”：通过寻找能访问的 CPU，然后列表，用户可以在列表中指定与哪个 CPU 进行连接。

4）单击菜单栏的“变量”→“触发器”，如图 3-47 所示，选择合适的触发点和触发频率。

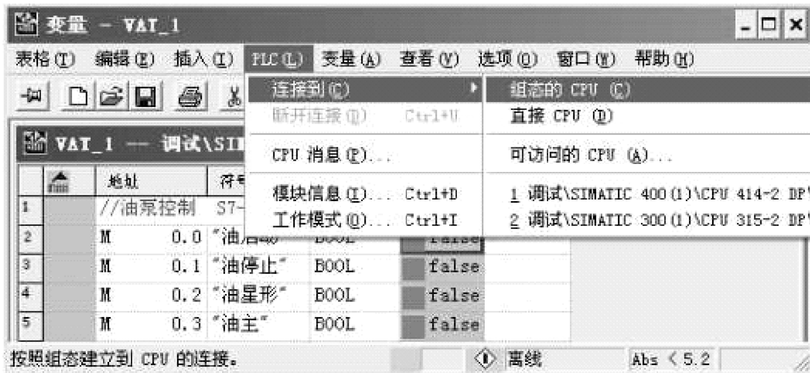


图 3-46 建立连接



图 3-47 设置触发器

在变量表进行修改和监视操作前，必须要设置触发器的触发点和触发条件，常见的选项功能如下：

触发点：通过指定触发点，可确定何时为要修改的变量分配固定值以及何时更新要监视的变量的值。

触发条件：通过指定触发频率，可确定是在到达触发点时仅分配变量值一次，还是每个扫描周期都分配；还可以确定是更新值一次，还是每个周期都更新。所以必须为修改和监视操作分别设置触发频率。

扫描循环开始：选择此选项是在扫描周期开始时分配或更新变量值。

扫描循环结束：选择此选项是在扫描周期结束时分配或更新变量值。

过渡到 STOP：选择此选项是在从 RUN 切换到 STOP 时分配或更新变量值。

一次：选择此选项是在到达触发点时仅分配或更新固定变量值一次。选择此项，会有一条消息通知是否可修改指定变量（例如，如果缺少相应的数据块，则不可修改数据字）。

每次循环：选择此选项是在每个扫描周期到达触发点时都分配或更新固定变量值。选择此项，不会输出任何消息通知是否可修改这些变量。

5）单击菜单栏的“变量”→“监视”和其他修改项目，打开、关闭监视和修改功能，如

图 3-48 所示。

监视：利用此菜单命令可显示在当前变量表中选定变量的监视值，其方式是根据已定义的触发点进行显示。快捷键是“”。

修改：利用此菜单命令可根据已定义的触发点为当前变量表中的选定变量分配固定值。快捷键是“”

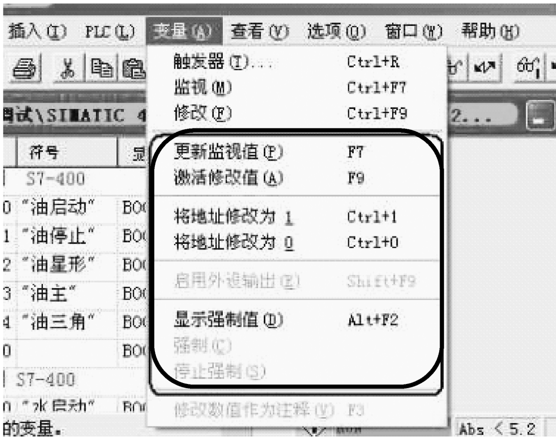


图 3-48　对变量进行操作选择

将地址修改为 1（或 0）：立即将表格中所有可见的选定地址（可选择单个或多个）都更改为值 1（或 0）。

启用外设输出：使用此菜单命令可以激活或取消激活“允许外围设备输出”模式。如果菜单命令旁边出现一个复选标记，并且呈灰色显示，则表示：

① 在其他“监视/修改变量”窗口中激活了“允许外围设备输出”模式，例如，由其他用户在其编程设备或 PC 上激活。

② 无法取消激活此模式，因为只能在激活此模式的窗口中取消激活该模式。

仅在 CPU 处于 STOP 模式时此功能才可用。与强制的外围设备输出相反，对于每个周期都修改的外围设备输出，在激活允许外围设备输出时并不会立即对相应的输出模块生效。

常用此项功能来检查可访问硬件配置中的所有模块（比如：I/O 模块测试，检查是否发生断路）。

显示强制值：利用此菜单命令可在“强制值”窗口中显示选定 CPU 变量的强制值。支持强制功能的 CPU 受到以下限制：

强制 S7-300 模块时仅允许强制输入和输出（I、Q）。

强制 S7-400 模块时仅允许强制外围设备的输入、外围设备的输出、标记（M）、输入和输出（I、Q）。

强制：利用此菜单命令可将在“强制值”窗口中输入的固定强制值分配给各个变量。强制变量和修改变量之间的差别很大，具体如表 3-7 所示。

表 3-7　强制变量和修改变量间的差别说明

特性/功能	S7-400 中的强制（包括 CPU318-2DP）	S7-300 中的强制（除 CPU318-2DP 外）	修改
标志(M)	是	—	是
定时器和计数器(T、C)	—	—	是
数据块(DB)	—	—	是
外围设备的输入(PIB、PIW、PID)	是	—	—
外围设备的输出(PQB、PQW、PQD)	是	—	是
输入与输出(I、Q)	是	是	是
用户程序可以覆盖修改/强制值	—	是	是

（续）

特性/功能	S7-400 中的强制（包括 CPU318-2DP）	S7-300 中的强制（除 CPU318-2DP 外）	修改
不必为中断有效时替换强制值	是	是	—
当应用程序退出后，变量值保持	是	是	—
当与 CPU 的连接中断后，变量值保持	是	是	—
允许的寻址错误： 例如 IW1 修改/强制值：1 IW1 修改/强制值：0	—	—	最后一个变为有效值
设置触发器	立即触发	立即触发	一次或每个周期
功能仅在激活窗口的可视区域影响变量	影响所有强制值	影响所有强制值	是

通过“启用外围设备的输出”，使强制外围设备输出的强制值在相应的输出模块上生效；然而，外围设备输出的修改值不会生效。

对于强制变量，变量始终具有强制值。每次读取用户程序时，都会读取该值。所有形式的写访问都无效。

对于永久的修改，程序的读取访问有效并一直持续到下一个触发点。

将修改/强制值作为注释命令（快捷键是“ ”）有两种情况：

将修改值作为注释激活或取消激活变量表中某变量或多个选定变量的修改值。

将强制值作为注释激活或取消激活“强制值”窗口中某变量或多个选定变量的强制值。

行无效（编辑功能）（符号是“ ”）：将变量表的当前行或选定行切换为无效/有效。通常使用此功能是将要选定行排除在监视和修改操作之外。

6）单击菜单栏的“表格”→“保存”或“另存为”来保存变量表，以便可以随时再次调用它。

3.5.2 修改变量

要求能够在程序状态中修改变量最起码的条件是打开在线块并进入监控状态。

1. 修改布尔数据类型的变量

1）选择想要修改的地址，如图 3-49 所示（我们此次修改 M0.5）。



图 3-49 选择布尔地址

2）单击菜单栏的“调试”→“修改地址为 1”或“修改地址为 0”，如图 3-50 所示。



图 3-50　修改布尔变量状态

2. 修改非布尔型变量

1）选择想要修改的地址。

2）单击菜单栏的“调试”→“修改地址”，如图 3-51 所示。



图 3-51　打开修改非布尔变量的界面

3）在弹出的对话框中，输入变量要采用的值（修改值为“s5t#30s”），如图 3-52 所示。



图 3-52　输入修改的数值

4）单击“确定”，关闭对话框。

3.6 测试程序

3.6.1 监视程序状态

1. 测试程序的原则

开始调试时，不要调试整个程序，要逐个地调用块，然后单独对调用的块进行调试。调试块的基本原则是：应当从调用体系最深的嵌套层的块开始，比如在 OB1 中调用它，然后通过监视和修改变量，为准备调试的块创建要测试的环境。如果要设置断点，以单步模式来调试程序，必须设置测试操作模式，这些测试功能不能用于过程操作模式。

2. 测试程序的步骤

（1）在线打开块

有项目管理操作步骤如下：

1）在 SIMATIC 管理器中打开项目的在线界面，如图 3-53 所示单击“ ”图标，即可打开在线界面（前提是编程设备必须与 PLC 连接而且可以通信）。



图 3-53 项目管理界面

2）从在线窗口中单击“块”文件夹，显示在线块的目录，如图 3-54 所示。

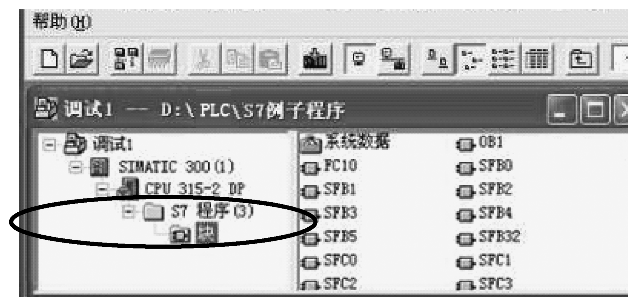


图 3-54 在线块中的目录

3）双击希望打开的块，比如双击图 3-54 中的“FC10”图标，打开 FC10 的在线界面，如图 3-55 所示。

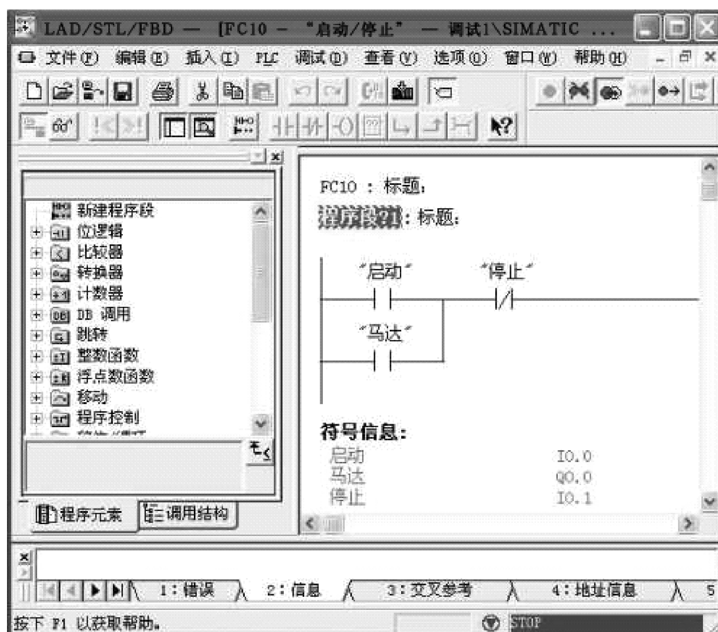


图 3-55　FC10 在线块界面

没有项目管理操作步骤如下：

1）在 SIMATIC 管理器中，单击“PLC”→“显示可访问的节点”，如图 3-56 所示。



图 3-56　打开显示可访问节点界面

2）从显示的列表中单击节点（“MPI = . . . ”对象），并单击“块”文件夹以显示可访问节点的列表，如图 3-57 所示。

在可访问节点的列表中，双击希望打开的块。

（2）设置程序状态的显示

可以在语句表、功能块图或梯形图中设置程序状态的显示。操作如下：

1）在程序块界面中，单击菜单栏的“选项”→“自定义”，如图 3-58 所示。

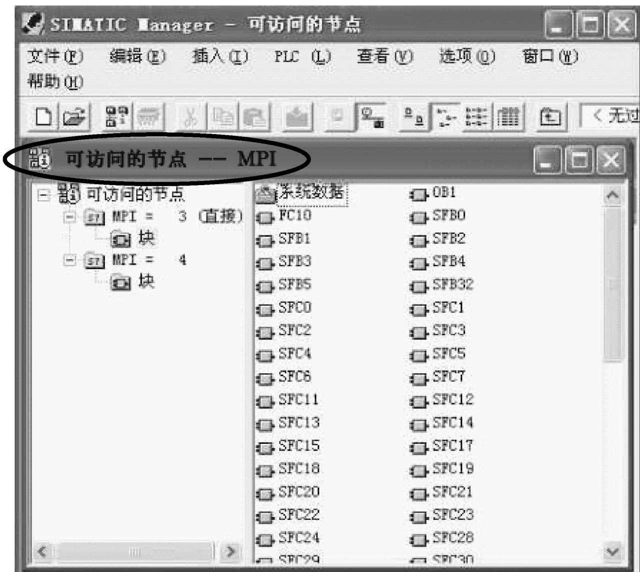


图 3-57　显示可访问节点的列表



图 3-58　打开自定义界面

2）在“自定义”对话框中，单击“STL”标签或“LAD/FBD”标签，如图 3-59 所示。

3）在“STL”标签或“LAD/FBD”标签中，选择用于测试程序所需的选项。在“STL”标签中可以选择显示见表 3-8。

表 3-8　程序状态可以显示的状态条目

可以激活的条目		说　明
英文	中文	
STA	状态位	状态位，即状态字的第 2 位
RLO	逻辑操作的结果	状态字的第 1 位，显示逻辑操作或算术比较的结果
STANDARD	默认状态	累加器 1 的内容

（续）

可以激活的条目		说　明
英文	中文	
AR1	地址寄存器 1	使用寄存器间接寻址的相关地址寄存器 1 的内容
AR2	地址寄存器 2	使用寄存器间接寻址的相关地址寄存器 2 的内容
ACC2	累计器 2	累加器 2 的内容
DB1	DB 寄存器 1	第一个打开的数据块寄存器的内容
DB2	DB 寄存器 2	第二个打开的数据块寄存器的内容
INDIRECT	间接	间接内存参考：指针参考（地址），没有地址内容参考，仅适用于内存间接寻址，不能用于寄存器间接寻址 若相应的指令出现在语句中时，可以显示定时器字或计数器字的内容
STATUS WORD	状态字	状态字中所有状态位的内容

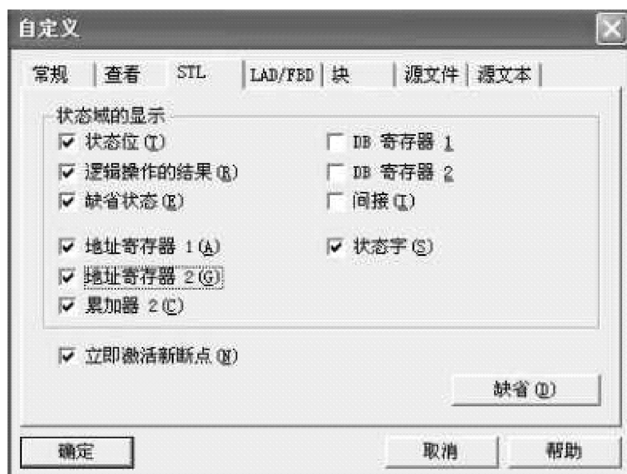


图 3-59　自定义界面



注意：当把 CPU315-2DP 作为从站，把 CPU315-2DP 作为主站时的诊断地址

在组态一个 CPU315-2DP 站时，你使用 S7 工具“H/WCONFIG”来分配诊断地址。如果发生一个故障，这些诊断地址被加入诊断 OB 的变量“OB82_MDL_ADDR”里。可以在 OB82 里分析此变量，确定有故障的站并做出相应的反应。

下面是如何分配诊断地址的例子：

第 1 步：通过 CPU315-2DP 组态从站并赋予一个诊断地址，比如 422。

第 2 步：通过 CPU315-2DP 组态主站。

第 3 步：把组态好的从站链接到主站并赋予一个诊断地址，比如 1022。

3.6.2　断点调试

要使用断点测试程序，必须满足：在线打开块；测试操作模式；在指令表（STL）中进行；不得设置保护块等几个条件。在断点测试中，断点工具栏常用到的快捷键命令如图 3-60 所示。

使用断点进行测试的步骤和方法表达如下：

在开始测试前，要确保 CPU 处于 RUN 模式或 RUN-P 模式，并且要测试的块已被保存并下载到 CPU。



图 3-60　断点测试快捷键

1）在项目管理界面，在在线块目录中打开准备测试的程序块。在项目管理界面，单击“ ”即可进入在线状态。

2）单击菜单栏的“调试”→“操作”，显示所设置的测试环境，在显示的对话框中选择模式：测试操作或过程操作，在这里选择测试模式。注意：当分配 CPU 参数时，如果设置了操作模式，只能通过改变参数来改变操作模式。

3）单击菜单栏的“查看”→“断点栏”，激活或隐藏断点工具栏。

4）把光标放在希望设置断点的语句行的前面，单击“ ”来设置断点。未激活的断点用空心圆圈标记，如图 3-61 所示。

5）单击“ ”来激活断点，空心圆圈标记变为实心圆圈标记，表示该断点被激活，同时自动弹出 PLC 寄存器内容显示界面，如图 3-62 所示。

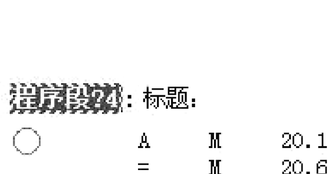


图 3-61　未激活的断点

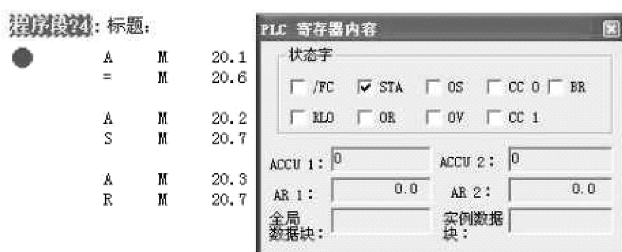


图 3-62　激活的断点

6）现在把 CPU 切换到 RUN-P。

7）当程序执行到断点时，CPU 转到 HOLD 模式。断点用箭头标记，如图 3-63 所示。



图 3-63　CPU 遇断点进入 HOLD 状态

8）要继续运行程序直到下一个断点，单击“●→”，将自动执行到下一个断点。

9）测试时要一句一句地执行程序，单击“ ”，在单步模式下进行测试。

10）如果有调用块存在，需要执行调用的块进行断点测试，可以单击“ ”即可进入调用块中执行断点测试。

11）可以单击“●”逐个删除断点，或者单击“ ”删除所有断点。

12）删除所有断点后，单击“●→”，即可恢复到 RUN 状态。

如果要显示调用顺序，需要在 SIMATIC 管理器中单击菜单栏的“PLC”→“诊断/设置”→“模块信息”。

如果要查看下一个断点，单击菜单栏的“显示下一个断点”。光标将跳转到所选择的下一个断点，而无需处理块。

在下载期间，拒绝下载在 PLC 中具有断点的块，只有在删除断点后才能下载。

第 4 章

S7-300/400指令功能及应用

4.1 PLC 程序概述

4.1.1 程序的组成与结构

PLC 的 CPU 运行两种程序，即系统程序和用户程序。

系统程序主要负责更新输入过程映像表，输出过程映像表；调用用户程序，采集中断信息，调用中断 OB；识别错误并进行错误处理；管理存储区域；与编程设备和其他通信设备进行通信等功能。

用户程序是用户为处理特定自动化任务所要求的功能编制的程序。用户程序完成的任务包括：处理过程数据；响应中断、处理正常程序周期中的干扰。

在 STEP 7 软件中，程序的基本单元是“块”。每个块是一个单个、独立的程序段。块间可以相互调用。调用块的类型不限，被调用块只能是功能块（OB 除外的其他块）。在 STEP 7 用户程序内可使用多种类型的块如：

1）组织块（OB）：确定用户程序的结构。

2）系统功能块（SFB）和系统功能（SFC）：SFB 和 SFC 集成在 S7 CPU 中，提供一些重要的系统功能。

3）功能块（FB）：FB 是带有用户可自行编程的“存储器”的块。

4）功能（FC）：FC 包含频繁使用的功能的例行程序。

5）数据块（DB）：DB 是用于存储用户数据的数据区。DB 分为背景 DB 和共享 DB。背景 DB 是属于特定的 FB 的，在创建时需要确定它所属的 FB。共享数据块为用户程序提供一个可保存的数据区，可由任何一个块来定义和使用。

OB、SFB、SFC、FB、FC 包含程序段，因此也称为逻辑块。每种块类型许可的块的数目和块的长度由 CPU 型号决定。

S7-300 PLC 程序各部分的关系如图 4-1 所示，本节我们只简单介绍，如需详细了解，请查看本书第 5 章内容。

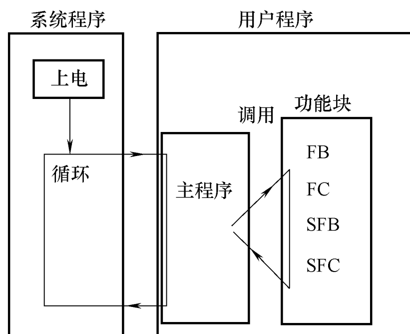


图 4-1　程序组成关系

主程序中，组织块（OB）确定单个程序段执行的顺序（启动事件）。一个 OB 调用可以中断另一个 OB 的执行。哪个 OB 允许中断另一个 OB 取决于其优先级。高优先级的 OB 可以中断低优先级的 OB。背景 OB 的优先级最低。

1. 功能块

功能块由两部分组成，一部分是变量声明表，用来声明这个块的局部数据（详见 4.1.2 节内容）；另一部分是指令代码组成的程序，程序使用变量表中定义的局部数据。

当要调用一个功能块时，要给它提供参数，称为参数传递。这些参数值赋给变量表中的变量，块程序执行时会使用它们。功能块具有通用性，不同的参数值会产生不同的运行结果。所以功能块可以被其他块调用，完成多个类似的任务。

变量声明表中的局部数据分为参数和局部变量两类，局部变量又分为静态变量和临时变量。参数指调用块与被调用块间传递的数据，局部变量是只供块本身使用的数据。FC 和 FB 是用户自己编写的程序块，它们都可以带有参数，参数类型见表 4-1。

表 4-1　FB 与 FC 的参数类型

参数类型	说　明
in	由调用块提供的数据输入
out	块执行后的返回参数,即输出结果
in_out	由调用块提供的输入数据,经逻辑块处理后再返回
temp	临时变量,存储在 L 堆栈中
stat	静态变量,存储在背景数据块中

FC 与 FB 的区别在于数据存储区。FB 拥有数据存储区，即背景 DB。而 FC 没有自己的数据区。就是说，在参数类型中，静态变量（stat）只能存在于 FB 中。功能块（FB）的参数存储在它的背景 DB 中，其内容可以保持，所以在调用时如果不给它的形式参数赋值，它会自动读取背景 DB 中的参数值。功能（FC）在调用时必须给其形式参数赋值。

系统功能（SFC）和系统功能块（SFB）是系统提供的编制好的 FC 和 FB，可供用户程序调用。SFC 和 SFB 固化在系统程序中，通常提供一些系统级的功能调用。

2. 组织块

组织块（OB）是操作系统和用户程序之间的接口。用户程序一般由启动程序、主程序和各种中断响应程序等模块组成，这些模块就是组织块。组织块由操作系统调用，控制循环、中断、驱动的程序执行以及 PLC 启动特性和错误处理等。可以对组织块进行编程来确定 CPU 的工作特性，不同型号的 CPU 支持不同的 OB。OB1 是一个特殊的组织块，操作系统在每次程序循环运行中，都会调用主程序中的组织块 OB1，即循环执行 OB1 中的用户程序，是 STEP 7 程序的主干部分。其他的 OB 对应启动程序、背景程序及各类中断处理程序。

由于组织块的调用是操作系统管理的，所以它只有定义在 L 堆栈中的临时变量。

4.1.2　变量编程及存储区

用户程序中的所有数据可分成 3 个类型，即

1）STEP7 提供的基本数据类型。

2）用户通过组合基本数据类型创建的复杂数据类型。

3）定义用来给 FB 或 FC 传送参数的参数类型。

1. 变量类型

(1) 基本数据类型

每种基本数据类型具有固定的长度。表 4-2 列出了基本数据类型。

表 4-2 基本数据类型

数据类型	位数	格式选项	范围和计数法	实例
BOOL(位)	1	布尔值	TRUE/FALSE	TRUE
BYTE(字节)	8	十六进制数	B#16#0 ~ B#16#FF	L B#16#10 L byte#16#10
WORD (字)	16	二进制数 十六进制数 BCD 码 十进制无符号数	2#0 ~ 2#1111_1111_1111_1111 W#16#0 ~ W#16#FFFF C#0 ~ C#999 B#(0.0) ~ B#(255.255)	L 2#0001_0000_0000_0000 L W#16#1000 L word#16#1000 L C#998 L B#(10,20) L byte#(10,20)
DWORD (双字)	32	二进制数 十六进制数 十进制无符号数	2#0 ~ 2#1111_1111_1111_ 1111_1111_1111_1111_1111 DW#16#0000_0000 ~ DW#16#FFFF_FFFF B#(0,0,0,0) ~ B#(255,255,255,255)	L 2#1000_0001_0001_1000_ 1011_1011_0111_1111 L DW#16#00A2_1234 L dword#16#00A2_1234 L B#(1,14,100,120) L byte#(1,14,100,120)
INT(整数)	16	十进制有符号数	-32 768 ~ 32 767	L 1
DINT(双整数)	32	十进制有符号数	L#-2147483648 ~ L#2147483647	L L#1
REAL (浮点数)	32	IEEE 浮点数	上限:±3.402 823e+38 下限:±1.175 495e-38	L 1.234567e+13
S5TIME (时间)	16	S7 时间,步长 10ms	S5T#0H_0M_0S_10MS ~ S5T#2H_46M_30S_0MS	L S5T#0H_1M_0S_0MS
TIME (时间)	32	IEC 时间,步长 1ms	-T#24D_20H_31M_23S_648MS ~ T#24D_20H_31M_23S_647MS	L T#0D_1H_1M_0S_0MS
DATE(日期)	16	IEC 日期,步长为 1 天	D#1990-1-1 ~ D#2168-12-31	L D#1996-3-15 L DATE#1996-3-15
TIME_OF_DAY (时间)	32	时间,步长为 1ms	TOD#0:0:0.0 ~ TOD#23:59:59.999	L TOD#1:10:3.3 L TIME_OF_DAY#1:10:3.3
CHAR(字符)	8	ASCⅡ字符	A、B 等	L 'E'

(2) 复杂数据类型

定义的是大于 32 位的数字数据群或包含其他数据类型的数据群。STEP 7 定义的复杂数据类型有：DATE_ AND_ TIME（日期和时间），STRING（字符串），ARRAY（数组），STRUCT（结构），UDT（用户自定义数据类型），FB 和 SFB（功能块和系统功能块）。表 4-3 中描述了这些复杂数据类型。其中有关“数据块”等“块”的概念已在 4.1.1 节详细说明。

表 4-3 复杂数据类型

数据类型	说　明
DATE_AND_TIME (日期和时间)	定义具有 64 位(8B)的区域,此数据类型以二进制编码的十进制格式保存
STRING (字符串)	定义最多有 254 个字符的字符串,为字符串保留的标准区域是 256B 长,包括 254 个字符和 2B 的标题所需要的空间,可以通过定义即将存储在字符串中的字符数目来减少字符串所需要的存储空间(例如:STRING[9]'Siemens')
ARRAY(数组)	定义一个数据类型(基本或复杂)的多维数组。例如:“ARRAY[1…2,1…3]OF INT”定义 2×3 的整数数组。使用下标(“[2,2]”)访问数组中存储的数据,最多可以定义 6 维数组,下标可以是任何整数(-32768 ~ 32767)
STRUCT(结构)	定义一个数据类型任意组合的结构体,例如,可以定义结构的数组或结构和数组的结构
UDT(用户自定义 数据类型)	在创建数据块或声明变量时,简化大量数据的结构化和数据类型的输入,在 STEP7 中,可以组合复杂的和基本的数据类型以创建用户的“用户自定义”数据类型,UDT 具有自己的名称,因此可以多次使用
FB 和 SFB(功能块和 系统功能块)	确定分配的背景数据块的结构,并允许在一个背景数据块中传送数个 FB 调用的背景数据

（3）参数类型

除了基本和复杂数据类型外，STEP 7 还允许为块之间传送的形式参数定义参数类型。STEP 7 可以定义下列参数类型。

1）TIMER 或 COUNTER：2B 长，指定当执行块时将使用的特定定时器或特定计数器。如果赋值给 TIMER 或 COUNTER 参数类型的形式参数，相应的实际参数必须是定时器或计数器，如 T1、C10。

2）块：2B 长，指定用作输入或输出的特定块。参数的声明确定使用的块类型（BLOCK_ FB、BLOCK_ FC、BLOCK_ DB、BLOCK_ SDB 等）。赋给 BLOCK 参数类型的形式参数，指定块地址作为实际参数，例如，“FC101”。

3）POINTER：6B 长，参考变量的地址。指针包含地址而不是值。当赋值给 POINTER 参数类型的形式参数，指定地址作为实际参数。在 STEP 7 中，可以用指针格式或简单地以地址指定指针。例如，M50.0，若寻址以 M50.0 开始的数据的指针，则定义为 P#M50.0。

4）ANY：10B 长。当实际参数的数据类型未知或当可以使用任何数据类型时，可以使用这种定义方式。如 P#M50.0 BYTE 10 即定义了数据类型的 ANY 格式。

参数类型也可以在用户自定义数据类型（UDT）中使用。

2. 存储区与变量的关系

CPU 的存储器可以划分为装载存储器、工作存储器以及系统存储器。

装载存储器用于存储用户程序，可以是 RAM 或 EPROM。未标记为启动时所需要的块将只存储在装载存储器中。

工作存储器（集成的 RAM）包含了与运行程序相关的部分 S7 程序。该程序仅在工作存储器和系统存储器中执行。

系统存储器（RAM）包含了每个 CPU 为用户程序提供的存储器单元，例如过程映像输入和输出表、位存储器、定时器和计数器。系统存储器也包含块堆栈和中断堆栈。CPU 的系统存储器还提供了临时存储器（局部数据堆栈），存放调用块时用到的临时数据。这些数据只在块激活时才保持有效。



注意：变量是如何存储在临时局部数据中的？

L 堆栈永远以地址“0”开始。在 L 堆栈中，会为每个数据块保留相同个数的字节，作为存放每个块所拥有的静态或局部数据。

当某个块终止时，那么它的空间随之也被重新释放出来。指针总是指向当前打开块的第一个字节。

除了上述的区域外，CPU 还有两个 32 位的累加器（ACCU1 和 ACCU2），两 个 地址寄存器 AR1 和 AR2，两个数据块地址寄存器 DB 和 DI，一个状态字寄存器。

状态字用来表示 CPU 执行程序时的一些重要状态，其结构如图 4-2 所示。

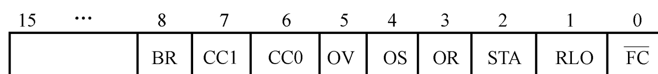


图 4-2　状态字的结构

（1）首次检测位$\overline{FC}$

首次检测是指 CPU 对逻辑串第一条指令的检测，检测结果保存在 RLO 位中。$\overline{FC}$位在逻辑串开始时为 0，在执行过程中变为 1。执行逻辑串结束的指令，如输出指令或与逻辑运算有关的转移指令，又将$\overline{FC}$清 0。

（2）逻辑操作结果 RLO

此位存储位逻辑运算及比较运算的结果。在逻辑串中，信号流的状态可以由 RLO 表示，有信号流时，RLO =1；没有信号流，RLO =0。

（3）状态位 STA

此位只在程序测试中由 CPU 使用。当位逻辑指令访问存储区时，STA 与被操作位的值是相同的。位逻辑指令不访问存储区时，STA 保持为 1。

（4）或位 OR

在先进行逻辑与运算，后进行逻辑或运算的逻辑串中，OR 用来暂时保存逻辑与的结果。其他指令执行时，OR 位清 0。

（5）存储上溢 OS

OV 被置 1 时，OS 也置 1，OV 被清 0 后 OS 还可以继续保持为 1。OS 的状态表示先前的指令执行是否出过错。块调用、块结束指令以及 JOS 指令能使 OS 清 0。

（6）溢出位 OV

当一个算术运算或浮点数比较运算执行时出错，其运算结果不正常时，OV 被置 1。

（7）条件码 CC0、CC1

算术和逻辑运算中，条件码用来表示累加器 1 中的运算内容与 0 的大小关系。在比较指令和移位指令中，条件码表示比较结果或移出位状态，见表 4-4 和表 4-5。

表 4-4 算术运算中的条件码

CC1	CC0	运算不溢出	整数算术运算溢出	浮点算术运算溢出
0	0	结果 =0	整数相加负溢出	下溢
0	1	结果 <0	乘法负溢出，加、减、取负正溢出	负溢出
1	0	结果 >0	乘、除正溢出，加、减负溢出	正溢出
1	1	—	除数为 0	非法操作

表 4-5 逻辑运算、比较运算及移位指令的条件码

CC1	CC0	逻辑运算	比较指令	移位指令
0	0	结果 =0	累加器 2 = 累加器 1	移出位 =0
0	1	—	累加器 2 < 累加器 1	—
1	0	结果 < >0	累加器 2 > 累加器 1	—
1	1	—	不规范	移出位 =1

（8）二进制结果位 BR

在既有位操作又有字操作的程序中，BR 可表示字操作结果是否正确。用户在编写 FB 和 FC 时，必须对 BR 位进行处理，功能块正确运行后，应使 BR =1，否则 BR =0。实现这种管理，可以用—(SAVE）指令，将 RLO 保存到状态字的 BR 位。功能块执行正确，使 RLO =1 并存入 BR，否则 RLO =0 并存入 BR。

3. 装载存储器和工作存储器

前面已有讲述，此处不再重复。

4. 系统存储器

系统存储器被划分成多个地址区，见表 4-6。使用指令，可以在相应的地址区域中直接

对数据寻址。

表 4-6　系统存储器结构

地址区	符号	功 能 描 述
过程映像输入表	I、IB、IW、ID	在扫描周期的开始，CPU 从输入模块读取输入，并记录该区域中的值
过程映像输出表	Q、QB、QW、QD	在扫描周期期间，程序计算输出值并将它们放入此区域。在扫描周期结束时，CPU 发送计算的输出值到输出模块
位存储器	M、MB、MW、MD	用于存储程序中计算的中间结果
定时器	T	为定时器提供存储空间
计数器	C	为计数器提供存储空间
数据块	DB、DBX、DBB、DBW、DBD 及 DI、DIX、DIB、DIW、DID	数据块包含程序的信息，它们可以被由所有逻辑块定义为通用（共享 DB），或者可以分配给特定的 FB 或 SFB（背景数据块）。用“OPEN DB”或“OPEN DI”打开
本地数据	L、LB、LW、LD	块的临时数据。或用于传送块参数和记录来自梯形图程序段的中间结果
外设输入区	PIB、PIW、PID	可直接访问的中央和分布式的输入模块（DP）
外设输出区	PQB、PQW、PQD	可直接访问的中央和分布式的输出模块（DP）

表中符号 I、Q 分别表示输入和输出。I、IB、IW、ID 分别表示输入位、输入字节、输入字和输入双字，其余类同。外设输入区和外设输出区的大小与 CPU 型号及具体系统配置有关，最大空间是 64KB。



注意：可以从 S7 CPU 中读出哪些标识数据？

通过 SFC51 “RDSYSST” 可读出下列标识数据：

可以读出订货号和 CPU 版本号。为此，使用 SFC51 和 SSLID0111 并使用下列索引：

1 = 模块标识

6 = 基本硬件标识

7 = 基本固件标识

过程映像输入/输出表是外设输入/输出区的前 128 个字节。CPU 既可间接使用过程映像表，也可直接通过底板/P 总线来访问集中和分布式数字输入/输出模块的输入和输出。当使用 STEP7 对模块进行组态时，需要将程序中使用的地址分配给模块。

对于集中 I/O 模块，在组态表中进行机架的分布以及模块到插槽的分配。

对于具有分布式 I/O（PROFIBUS DP 或 PROFINET IO）的工作站，在“主站系统”组态表中按 PROFIBUS 地址排列 DP 从站，并将模块分配给插槽。



注意：为 S7 CPU 上的 I/O 模块（集中式或者分布式的）分配地址时应当注意哪些问题？

请注意，创建的数据区域（如一个双字）不能组态在过程映像的边界上，因为在该数据块中，只有边界下面的区域能够被读入过程映像，因此不可能从过程映像访问数据。因此，这些组态规则不支持这种情况：例如，在一个 256B 输入的过程映像的 254 号地址上组态一个输入双字。如果一定需要如此选址，则必须相应地调整过程映像的大小（在 CPU 的 Properties 中）。

通过对模块进行组态，就不必再使用开关来设定每个模块上的地址。组态完成后，编程设备把数据发送给 CPU，从而使该 CPU 能够识别为其分配的模块。

与直接访问输入/输出模块相比，访问过程映像的主要优点在于在一个程序周期持续期间，CPU 具有过程信号的一致性的映像。如果在程序执行期间，输入模块的信号状态发生了变化，过程映像中的信号状态仍被保持，直到下一个周期过程映像进行了更新。在用户程序中周期性地扫描输入信号的过程，确保了总有一致的输入信息。访问过程映像还比直接访问信号模块速度更快。

本地数据区也叫局部数据堆栈或 L 堆栈。当对组织块编程时，可以声明临时变量（TEMP）只在块执行期间可用，然后它将被覆盖。在首次访问局部数据堆栈之前，必须对局部数据初始化。除此之外，每个组织块还需要 20B 的局部数据来存储它们的启动信息。

CPU 只能为当前执行的块的临时变量（局部数据）提供有限的存储空间。该存储器局部数据堆栈的大小取决于 CPU。局部数据堆栈被各优先级均分（默认）。也就是说每个优先级都有它自己的局部数据区，从而保证了较高的优先级和它们的 OB 自身的局部数据有可用的空间。

5. 定义符号

在程序中，访问 I/O 信号、位存储器、计数器、定时器、数据块和功能块等，可以使用绝对地址，也可以使用地址符号。

绝对地址包含地址标识符和存储器位置，例如，Q4.0、I1.1、M2.0、FB21 等。可以给绝对地址分配一个符号名，采用具有某种意义的符号名来代替绝对地址，比如把 MOTOR_ON 分配给 Q4.0，而这将使程序更容易阅读。

使用符号时要注意共享符号（全局符号）和局部符号的区别。共享符号在整个用户程序中有效，并且是唯一的，它在符号表中定义。局部符号在块的变量声明表中定义，并且只在定义的块中有效。

4.1.3 指令符号及寻址方式

梯形图指令是一种用图形符号表示的指令系统，它源于早期的继电器控制电路图，因为编写的程序像一级一级的梯子，故名“梯形图”。

在梯形图中，使用抽象的触点和线圈表示输入和输出，最左边的竖线称为母线，假想的电流从母线流出，经过一系列触点，控制右边的线圈，线圈通电后可以触发相关的触点，这和继电器控制原理相同。

1. 指令符号

常用的梯形图指令可分为 5 类，分述如下。

（1）继电器类

1）动合触点和动断触点：—| |—和—| / |—，在触点上面可以标明数据地址。动合触点表示在未触发时，触点是断开的，触发后接通；动断触点则相反。

2）输出线圈和中间输出：—( )和—(#)—。这是最常用的一类指令，直接来源于继电器控制电路。后来增加的功能指令，往往是在触点或线圈上加入文字注释生成的，如复位线圈指令—(R)，是在输出线圈的括号内加字母 R。又如在动合触点符号中加入“NOT”，即—| NOT |—，表示取反操作。

（2）定时、计数类

1）设置计数器值：—(SC)。

2）接通、断开延时定时器线圈：—(SD) 和—(SF)。

3）扩展脉冲定时器线圈：—(SE)。

4）脉冲定时器线圈：—(SP)。

5）保持接通延时定时器线圈：—(SS)。

这类指令很多只有助记符，而没有梯形图的图形符号。指令中用 S_CD 表示减计数器，S_CU 表示增计数器，S_CUD 表示双向计数器。

（3）算术运算类

包括加、减、乘、除、平方、开平方、对数、三角函数以及数制转换指令等。

（4）逻辑运算与比较类

简单的逻辑运算可以通过连线关系表示，如图 4-3 所示，满足下列条件之一时，将会触发输出 Q0.3：输入端 I0.0 和 I0.1 的信号状态为“1”时；或输入端 I0.2 的信号状态为“0”时。

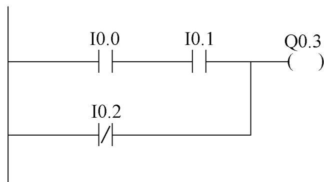


图 4-3　线“与”关系

在第一梯级，I0.0 和 I0.1 是逻辑“与”的关系，第一梯级和第二梯级是逻辑“或”的关系。另外，对整数、实数等都有比较指令。

（5）程序控制类

包括逻辑控制指令、程序调用与返回和主控继电器指令等 3 类。

2. 寻址方式

寻址方式是指获得操作数的方式，STEP 7 规定了 4 种寻址方式：立即寻址、直接寻址、存储器间接寻址和寄存器间接寻址。

（1）立即寻址

对常数和常量的寻址方式，操作数本身包含在指令中。有些指令的操作数是唯一的，指令中不写出来。在梯形图指令中，立即数直接写出。

（2）直接寻址

直接寻址是给出操作数的存储单元地址，对存储器和寄存器都可以直接寻址。

（3）存储器间接寻址

存储器间接寻址指令中给出的存储器，称为存储器指针，该存储单元的内容是操作数的存储地址。根据地址的长度，地址指针可以是字或双字。对于定时器、计数器、数据块和功能块，其编号小于 65 535，用字指针即可；而对于其他的地址，则要用到双字指针。用双字指针访问字节、字或双字存储器，指针中的位编号应为 0。

（4）寄存器间接寻址

S7-300 的 CPU 中有两个地址寄存器，AR1 和 AR2。通过它们，可以对各存储器实现寄存器间接寻址访问。地址寄存器的地址指针有两种格式，都是双字的。第一种地址指针适合作区内寄存器间接寻址，在指针中包含被寻址数值所在存储单元地址的字节编号和位编号，但指向哪个存储器，在指令中给出。第二种地址指针除了第一种的内容外，还具有存储器的标识位，通过改变这些标识位，可实现跨区寻址。

用寄存器指针访问字节、字或双字存储器，指针中的位编号应为 0。

4.2　位逻辑指令

4.2.1　位逻辑运算指令

位逻辑指令使用“1”和“0”两个数字。这两个数字组成了二进制数字系统的基础，称作二进制数字或位。在触点和线圈符号中，“1”表示激活或激励状态，“0”表示未激活或未激励状态。

位逻辑指令对“1”和“0”信号状态加以解释，并按照布尔逻辑组合它们。这些组合会产生由“1”或“0”组成的结果，称为逻辑运算结果（RLO）。由位逻辑指令触发的逻辑运算可以执行各种逻辑运算功能。

位逻辑指令中常用的符号是继电器类符号及定时器/计数器符号，如动合/动断触点、输出线圈、信号取反等，详见本书 4.1.3 节。另外还有 SAVE 指令、上升及下降沿检测指令等。由于梯形图指令是一种图形化指令，逻辑运算关系往往以类似电路连接的方式表达，而没有文本指令的逻辑“与”运算指令、逻辑“或”运算指令等形式，如图 4-3 中所示关系。对前面已给出的简单的符号指令，不再说明。

（1）SAVE 指令

SAVE 指令将 RLO 保存到状态字的 BR 位。使用时，应注意在退出块前使用 SAVE 指令，因为块执行期间，有许多指令可能会对 BR 位进行修改。

【例】　图 4-4 所示的例子中，一个梯级的运行结果被保存到了 BR 位。

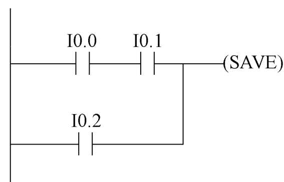


图 4-4　SAVE 指令

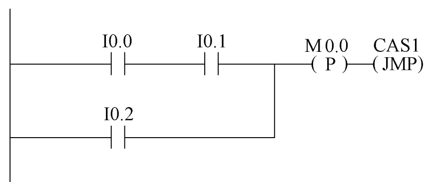


图 4-5　上升沿检测

（2）RLO 上升/下降沿检测

指令—(P)—和—(N)—检测地址中信号的变化，把上升沿/下降沿的变化，在指令后将其显示为 RLO =“1”/“0”。将 RLO 中的当前信号状态与地址的信号状态（边沿存储位）进行比较。如果在执行指令前地址的信号状态为“0”，RLO 为“1”，则在执行指令后 RLO 将是“1”，在所有其他情况下将是“0”。指令执行前的 RLO 状态存储在地址中。

【例】　图 4-5 所示是一个上升沿检测的程序。

边沿存储位 M0.0 保存 RLO 的先前状态。RLO 的信号状态从“0”变为“1”时，程序将跳转到标号 CAS1。

（3）置位/复位优先型双稳态触发器

对置位优先型双稳态触发器，如果 R 输入端的信号状态为“1”，S 输入端的信号状态为“0”，则复位 RS。否则，如果 R 输入端的信号状态为“0”，S 输入端的信号状态为“1”，则置位触发器。如果两个输入端的 RLO 均为“1”，触发器先在指定地址执行复位指

令，然后执行置位指令，以使该地址在执行余下的程序扫描过程中保持置位状态。

对复位优先型双稳态触发器，如果两个输入端的 RLO 均为“1”，触发器先在指定地址执行置位指令，然后执行复位指令，使该地址在执行余下的程序扫描过程中保持复位状态。只有在 RLO 为“1”时，才会执行 S（置位）和 R（复位）指令。RLO 为“0”时，指令中指定的地址保持不变。

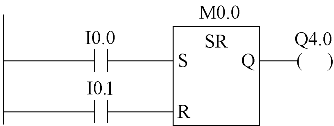


图 4-6　置位/复位指令

表 4-7 的指令中符号“RS”是表示置位优先型，如果是“SR”，则表示复位优先型。图 4-6 所示是置位/复位触发器指令的使用。

表 4-7　置位/复位优先型 RS 双稳态触发器指令

LAD 指令	参数	数据类型	存储区	说　明
<地址> RS S Q R	<地址>	BOOL	I、Q、M、L、D	要置位或复位的位
	S			启用置位指令
	R			启用复位指令
	Q			<地址>的信号状态

如果输入端 I0.0 的信号状态为“1”，I0.1 的信号状态为“0”，则置位存储器位 M0.0，输出 4.0 将是“1”。否则，如果输入端 I0.0 的信号状态为“0”，I0.1 的信号状态为“1”，则复位存储器位 M0.0，输出 Q4.0 将是“0”。如果两个信号状态均为“0”，图 4-6 置位/复位指令则不会发生任何变化。如果两个信号状态均为“1”，将因顺序关系执行复位指令，复位 M0.0，Q4.0 将是“0”。

（4）立即读取

对于对时间要求苛刻的应用程序，对数字输入的当前状态的读取可能要比正常情况下每 OB1 扫描周期一次的速度快。立即读取在扫描“立即读取”梯级时从输入模块中直接获取数字输入的状态，不必等到下一 OB1 扫描周期结束。要从输入模块立即读取一个输入（或多个输入），必须使用外设输入（PI）存储区来代替输入（I）存储区。可以字节、字或双字形式读取外设输入存储区。因此，不能通过触点（位）元素读取单一数字输入。

立即读取是要满足一定条件的：CPU 读取包含相关输入数据的 PI 存储区的字；如果输入位处于接通状态（为 1），将对 PI 存储区的字与某个常数执行产生非零结果的 AND 运算；测试累加器的非零条件。

对于“立即读取”功能，必须按以下实例所示创建符号程序段。

【例】　图 4-7 所示是立即读取的程序。

对 WAND_W 指令的说明：

PIW1 = 0000000000101010

W#16#0002 = 0000000000000010

结果 0000000000000010

在此实例中，立即输入 I1.1 与 I4.1 和 I4.5 串联。

字 PIW1 包含 I1.1 的立即状态。对 PIW1 与 W#16#0002 执行 AND 运算。如果 PB1 中的 I1.1（第二位）为真（“1”），则结果不等于零。如果 WAND_W 指令的结果不等于零，触

点 A < >0 时将传递电压。程序中必须指定 MWx * 才能存储程序段。x 可以是允许的任何数。

（5）立即写入

立即写入与立即读取类似。要将一个输出（或多个输出）立即写入输出模块，要使用外设输出（PQ）存储区来代替输出（Q）存储区。可以字节、字或双字形式写入外设输出存储区。因此，不能通过线圈单元更新单一数字输出。要立即向输出模块写入数字输出的状态，根据条件把包含相关位的 Q 存储器的字节、字或双字复制到相应的 PQ 存储器（直接输出模块地址）中。

【例】 图 4-8 所示是立即写入的程序。

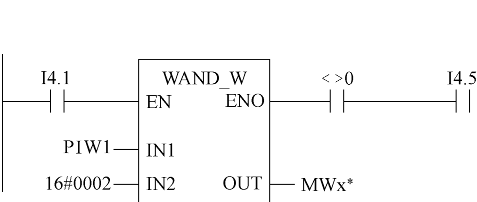


图 4-7　立即读取

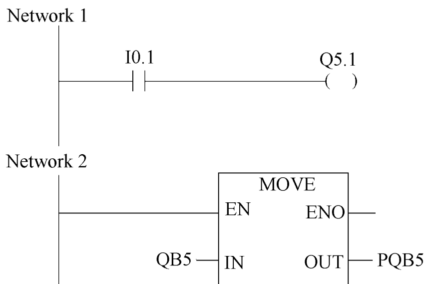


图 4-8　立即写入

在此实例中，Q5.1 为所需的立即输出位。

可以修改寻址输出 Q 字节（QB5）的状态位，也可以将其保持不变。程序段 1 中给 Q5.1 分配 I0.1 信号状态。将 QB5 复制到相应的直接外设输出存储区（PQB5）。

4.2.2　比较指令

比较指令是对两个操作数进行比较，比较关系有等于、不等于、大于、小于、大于等于和小于等于。如果比较结果为“真”，则函数的 RLO 为“1”。如果以串联方式使用比较单元，则使用“与”运算将其链接至梯级程序段的 RLO；如果以并联方式使用该单元，则使用“或”运算将其链接至梯级程序段的 RLO。按照操作数的类型可分为整数比较指令、双精度整数比较指令和实数比较指令 3 种，如图 4-9 所示。除数据类型不同外，这 3 种指令的符号和功能都一样。

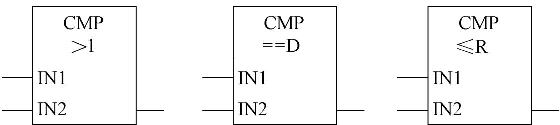


图 4-9　整数比较、双精度整数比较、实数比较指令

比较指令的使用方法与标准触点类似。它可位于任何可放置标准触点的位置，可根据用户选择的比较类型比较 IN1 和 IN2。

4.3　定时器、计数器指令

4.3.1　定时器指令

在 CPU 的存储器中，有一个区域是专为定时器保留的，此存储区域为每个定时器地址保留一个 16bit 字。梯形图逻辑指令集支持 256 个定时器，但每个具体的 CPU 型号支持的定时器个数不同。

定时器字的 0～9 位包含 BCD 码的时间值，或用 0～11 位存储二进制定时值。定时时间等于定时值乘以时间基准。时间更新操作按以时间基准指定的时间间隔，将时间值递减一个单位，直至时间值等于 0，定时时间到，此时定时器触点动作。可以用二进制、十六进制或以二进制编码的十进制（BCD）格式，将时间值装载到累加器 1 的低位字中。

定时器字的 12 和 13 位包含二进制编码的时间基准，其取值“00”、“01”、“10”、“11”，对应的时间基准是 10ms、100ms、1s 和 10s。时间基准越小，定时器分辨率越高，但定时范围会减小，详细关系请参看表 4-8。

表 4-8　时基设置与定时范围

时基设置	时间基准	定时范围
00	10ms	10ms～9s_990ms
01	100ms	100ms～1min_39s_900ms
10	1s	1s～16min_39s
11	10s	10s～2h_46min_30s

定时器的种类有脉冲定时器、扩展脉冲定时器、接通延时定时器、保持接通延时定时器和断开延时定时器。定时器指令有梯形图和简化指令（见附录）两种形式。

指令中 T no. 是定时器号，S 是启动输入端，上升沿有效。TV 是预设置的时间值。R 是复位输入端，R 从“0”变为“1”时，定时器将被复位，则当前时间和时间基准也被设置为 0。如果定时器不是正在运行，则定时器 R 输入端的逻辑“1”没有任何作用。Q 是定时器状态输出端，定时器运行时，Q＝1。BI 和 BCD 是剩余时间值输出端，分别提供整数和 BCD 码形式的值。下面给出几个使用定时器的例子。

【例】　图 4-10 所示是脉冲定时器的使用。

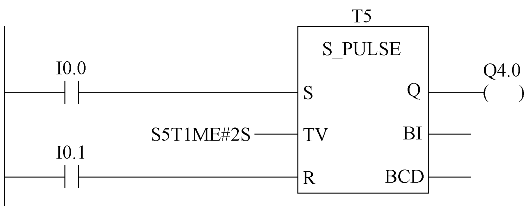


图 4-10　脉冲定时器指令

如果输入端 I0.0 的信号状态从“0”变为“1”（RLO 中的上升沿），则定时器 T5 将启动。只要 I0.0 为“1”，定时器就将继续运行指定的 2s。如果定时器达到预定时间前，I0.0 的信号状态从“1”变为“0”，则定时器将停止。如果输入端 I0.1 的信号状态从“0”变为“1”，而定时器仍在运行，则时间复位。

只要定时器运行，输出端 Q4.0 就是逻辑“1”，如果定时器预设时间结束或复位，则输出端 Q4.0 变为“0”。

扩展脉冲定时器与普通脉冲定时器有所不同。扩展脉冲定时器启动后，定时时间未到时，即使 S 端变为“0”，定时器仍然继续工作。如果在定时器运行期间输入端 S 的信号状态从“0”变为“1”，则将使用预设的时间值重新启动定时器。

对接通延时定时器，只要输入端 S 的信号状态为正，定时器就以在输入端 TV 指定的时间间隔运行。定时器达到指定时间而没有出错，并且 S 端的信号状态仍为“1”时，Q 端的信号状态为“1”。若定时器运行期间输入端 S 的信号状态从“1”变为“0”，定时器将停止，同时输出端 Q 的信号状态为“0”。如果在定时器运行期间复位（R）输入从“0”变为“1”，则定时器复位。当前时间和时间基准被设置为零。然后，输出端 Q 的信号状态变为“0”。如果在定时器没有运行时 R 输入端有一个逻辑“1”，并且输入端 S 的 RLO 为“1”，则定时器也复位。

保持接通延时定时器与延时定时器有所不同。它启动后，定时时间未到时，即使 S 端变为“0”，定时器仍然继续工作。如果在定时器运行期间输入端 S 的信号状态从“0”变为“1”，则将使用预设的时间值重新启动定时器。如果复位（R）输入从“0”变为“1”，则无论 S 输入端的 RLO 如何，定时器都将复位，该关系详见图 4-11 所示。然后，输出端 Q 的信号状态变为“0”。

【例】 保持接通延时定时器的例子如图 4-12 所示。

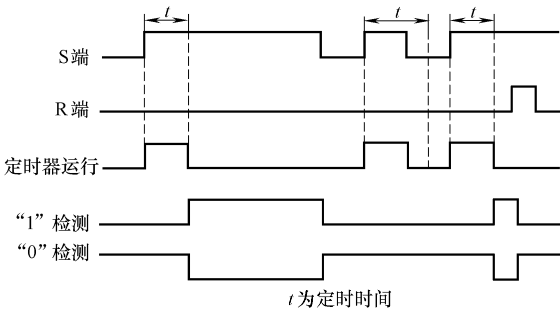


图 4-11 接通延时定时器的时序

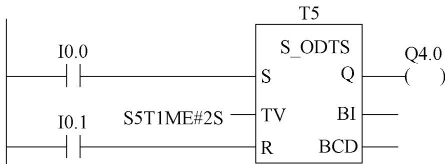


图 4-12 保持接通延时定时器指令

如果 I0.0 的信号状态从“0”变为“1”（RLO 中的上升沿），则定时器 T5 将启动。无论 I0.0 的信号是否从“1”变为“0”，定时器都将运行。如果在定时器达到指定时间前，I0.0 的信号状态从“0”变为“1”，则定时器将重新触发。如果定时器达到指定时间，则输出端 Q4.0 将变为“1”。如果输入端 I0.1 的信号状态从“0”变为“1”，则无论 S 处的 RLO 如何，时间都将复位，如图 4-13 所示。

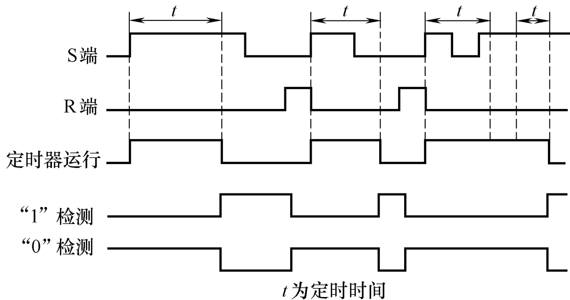


图 4-13 保持接通延时定时器的时序

对于断开延时定时器，如果 S 输入端的信号状态为“1”，或定时器正在运行，则输出端 Q 的信号状态为“1”。如果在定时器运行期间输入端 S 的信号状态从“0”变为“1”时，定时器将复位。输入端 S 的信号状

态再次从“1”变为“0”后，定时器才能重新启动。如果在定时器运行期间复位（R）输入从“0”变为“1”时，定时器将复位，如图 4-14 所示。

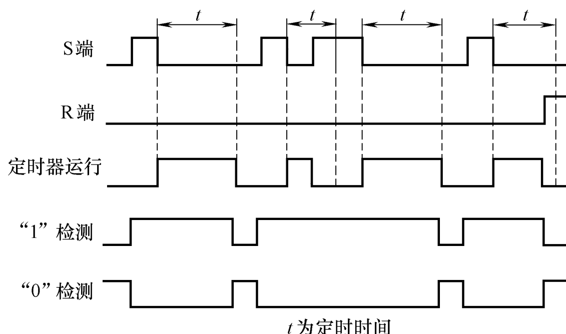


图 4-14　断开延时定时器的时序

4.3.2　计数器指令

计数器与定时器类似，该指令是用来访问计数器存储区的函数。在用户 CPU 的存储器中，为计数器保留有存储区。此存储区中为每个计数器地址保留一个 16bit 字。梯形图指令集支持 256 个计数器。

计数器是对 RLO 的上升沿计数的，它由计数器字和状态位组成。计数器字中的 0～9 位以二进制代码形式存储当前的计数值，或者用 0～11 位以 BCD 码形式存储当前的计数值。当设置某个计数器时，计数值移至计数器字。计数值的范围为 0～999。S7 有 3 种计数器，分别是加计数器、减计数器和双向计数器。

计数器指令有梯形图指令和简化指令（见附录）两种形式，图 4-15 所示是梯形图形式的指令。指令中 C no. 是计数器号，CU、CD 分别是加计数输入和减计数输入。PV 是计数器初始值，如设置计数初值为 120，用 BCD 码形式可写为 C#120。S 是计数预置输入端，R 是计数器复位端，Q 表示计数状态，CV 和 CV_BCD 是当前计数值的整数表示和 BCD 码表示。

下面以双向计数器说明计数器的使用。

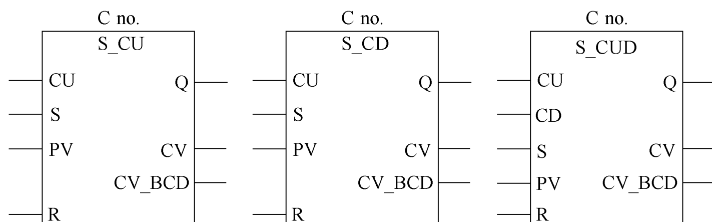


图 4-15　加、减、双向计数器的梯形图指令

如果输入端 S 有上升沿，S_CUD（双向计数器）预置为 PV 的输入值。如果输入端 R 为“1”，则计数器复位，并将计数值设置为 0。如果输入端 CU 的信号状态从“0”切换为“1”，并且计数器的值小于“999”，则计数器的值增 1。如果输入端 CD 有上升沿，并且计数器的值大于“0”，则计数器的值减 1。

如果两个计数器输入都有上升沿，则执行两个指令，而计数值保持不变。

如果已设置计数器，并且输入端 CU/CD 的 RLO＝1，即使没有从上升沿到下降沿或下降沿到上升沿的切换，计数器也会在下一个扫描周期进行相应的计数。

如果计数值大于等于“0”，则输出端 Q 的信号状态为“1”。图 4-16 所示是双向计数器的使用方法。

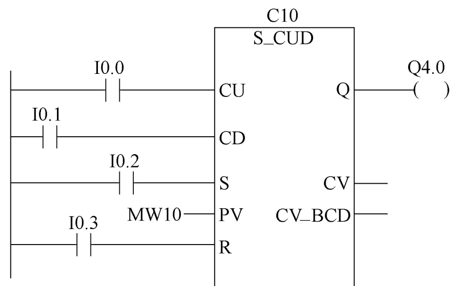


图 4-16　双向计数器的使用

如果 I0.2 从“0”变为“1”，则计数器预设为 MW10 的值。如果 I0.0 的信号状态从“0”变为“1”，则计数器 C10 的值将增加 1，除非 C10 的值等于 999。如果 I0.1 从“0”改变为“1”，则 C10 减少 1，除非 C10 的值为“0”。如果 C10 不等于 0，则 Q4.0 为“1”。

在 CPU 经过完全复位后是否运行时间计数器也被复位?



注意：在 CPU 经过完全复位后是否运行时间计数器也被复位?

使用 S7-300 时，带硬件时钟（内置的“实时时钟”）和带软件时钟的 CPU 之间有区别。对于那些无后备电池的软件时钟的 CPU，运行时间计数器在 CPU 被完全复位后其最后值被删除。而对于那些有后备电池的硬件时钟的 CPU，运行时间计数器的最后值在 CPU 被完全复位后被保留下来。同样，CPU318 和所有的 S7-400CPU 的运行时间计数器在 CPU 被完全复位后其最后值被保留。

4.4 数据处理指令与逻辑控制指令

4.4.1 数据处理指令

数据处理指令包括数据传送、转换、算术运算、循环移位及字逻辑指令。

转换指令读取输入参数 IN 的内容，然后进行数据类型转换或改变其符号。可通过参数 OUT 查询结果，具体指令见表 4-9。这一类指令相对比较简单，举例介绍如下：

【例】 如图 4-17 所示是一个 BCD 码转换成整数的例子。

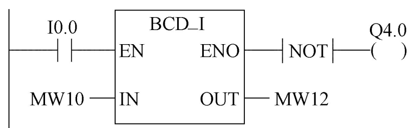


图 4-17 BCD 码转换成整数

如果输入 I0.0 的状态为“1”，则将 MW10 中的内容以 3 位 BCD 码数字读取，并将其转换为整型值，结果存储在 MW12 中。如果未执行转换（ENO = EN =0），则输出 Q4.0 的状态为“1”。其他各个转换指令与此类似。指令中参数 EN 和 ENO 是 BOOL 型，参数 IN 是待转换数据的类型，参数 OUT 是转换后的数据类型。

表 4-9 数据转换指令表

指令名	说明	指令名	说明
BCD_I	BCD 码转换为整数	I_BCD	整型转换为 BCD 码
BCD_DI	BCD 码转换为双精度整数	I_DINT	整型转换为长整型
DI_BCD	长整型转换为 BCD 码	DI_REAL	长整型转换为浮点型
INV_I	二进制反码整型	INV_DI	二进制反码长整型
NEG_I	二进制补码整型	NEG_DI	二进制补码长整型
NEG_R	浮点数取反	ROUND	取整为长整型
TRUNC	截断长整型部分	CEIL	向上取整
FLOOR	向下取整		

4.4.2　数据传输指令

1. 打开数据块指令

本指令用来打开共享 DB 或背景 DB。—(OPN) 函数是一种对数据块的无条件调用。将数据块的编号传送到 DB 或 DI 寄存器中。后续的 DB 和 DI 命令根据寄存器内容访问相应的块。指令说明见表4-10。

表4-10　打开数据块指令说明

指令	参数	数据类型	存储区	说明
(DB号)—(OPN)	DB号	BLOCK_DB	DB、DI	DB/DI编号范围取决于CPU型号

【例】 图4-18中，打开数据块10（DB10）。触点地址（DBX0.0）引用包含在DB10中的当前数据记录的数据字节0的第0位。将此位的信号状态分配给输出Q4.0。

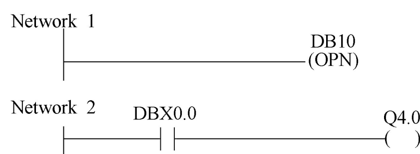


图4-18　打开数据块指令

2. 传送指令

MOVE 指令通过启用 EN 输入来激活。在 IN 输入的值将复制到在 OUT 输出的指定地址。ENO 与 EN 的逻辑状态相同。MOVE 只能复制 BYTE、WORD 或 DWORD 数据对象。

用户自定义数据类型（如数组或结构）必须用系统功能“BLKMOVE”（SFC 20）来复制。将某个值传送给不同长度的数据类型时，会根据需要截断或以0填充高位字节。

在激活的 MCR 区内，如果开启了 MCR，同时有通往启用输入的电流，则按如上所述复制寻址的数据。如果 MCR 关闭，则无论当前 IN 状态如何，执行 MOVE 均会将逻辑“0”写入到指定的 OUT 地址。

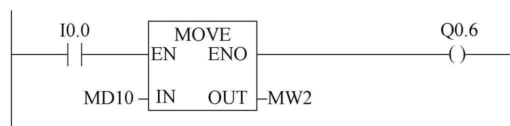


图4-19　传送指令

在图4-19所示传送指令中，传送指令功能块的参数说明如表4-11所示。

当 I0.0=1 时，MOVE 指令把 IN 输入指定的值复制到 OUT 输出指定的地址中。ENO 与 EN 的逻辑状态相同。

表4-11　传送指令功能块的参数

参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、L、D	启用输入
ENO	BOOL	I、Q、M、L、D	启用输出
IN	8位、16位或32位的基本数据类型	I、Q、M、L、D、常数	源
OUT	8位、16位或32位的基本数据类型	I、Q、M、L、D	目标



注意：在执行 MOVE 指令时，IN 与 OUT 不进行数值长度检查，将某个值传送给不同长度的数据类型时，会根据需要截断或以零填充高位字节，如表4-12实例所示。

表 4-12　IN 与 OUT 数据长度不同

IN	OUT
MB10 = 1111 0000	MB20 = 1111 0000
	MW20 = 0000 0000　1111 0000
	MD20 = 0000 0000 0000 0000　0000 0000　1111 0000
MD10 = 1111 1111 0000 1111 1111 0000 0101 0101	MB20 = 0101 0101
	MW20 = 1111 0000　0101 0101
	MD20 = 1111 1111 0000 1111 1111 0000 0101 0101

4.4.3　状态字指令

状态位指令用于对状态字的位进行处理。各状态位指令分别对下列条件之一做出反应，其中每个条件以状态字的一个或多个位来表示。

1）二进制结果位被置位（BR —|　|—），即 信 号 状态为 1。

2）数学运算指令发生溢出（OV —|　|—）或存储溢出（OS —|　|—）。

3）算术运算功能的结果无序（UO —|　|—）。

4）数学运算函数的结果与 0 的关系有：

= =0、< > 0、> 0、< 0、≥0、≤0。

当状态位指令以串联方式连接时，该指令将根据与真值表将其信号状态校验的结果与前一逻辑运算结果合并。当状态位指令以并联方式连接时，该指令将根据“或”真值表将其结果与前一 RLO 合并。

1. 异常位溢出

指令—| OV |—（异常位溢出）或—| OV / |—（异常位溢出取反）触点符号用于识别上次执行数学运算指令时的溢出。检查指令执行后的结果是否超出了允许的正、负范围。串联使用时，扫描的结果将通过 AND 与 RLO 连接；并联使用时，扫描结果通过 OR 与 RLO 连接。

【例】　在图 4-20 所示程序中，I0.0 的信号状态为“1”时将激活减法运算。相减的结果超出了整数的允许范围，则置位 OV 位。如果 OV 扫描的信号状态为“1”且程序段 2 的 RLO 为“1”，则置位 Q4.0。

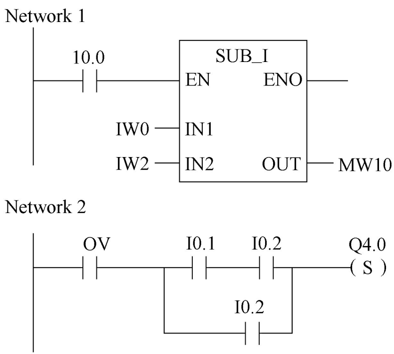


图 4-20　溢出位检测

2. 存储的异常位溢出

—| OS |—（存储的异常位溢出）或—| OS / |—（存储的异常位溢出取反）触点符号用于识别和存储数学运算函数中的锁存溢出。如果指令的结果超出了允许的负或正范围，则置位状态字中的 OS 位。与需要在执行后续数学运算函数前重写的 OV 位不同，OS 位在溢出发生时存储。OS 位将保持置位状态，直至离开该块。串联使用时，扫描的结果将通过 AND 与 RLO 连接；并联使用时，扫描结果通过 OR 与 RLO 连接。

3. 无序异常位

—| UO |—（无序异常位）或—| UO / |—（无序异常位取反）触点符号用于识别含浮点数

的数学运算是否无序（也就是说，数学运算函数中的值是否有无效浮点数）。如果含浮点数的数学运算函数的结果无效，则信号状态扫描为“1”。如果 CC1 和 CC0 中的逻辑运算显示“无效”，信号状态扫描的结果将是“0”。串联使用时，扫描的结果将通过 AND 与 RLO 连接；并联使用时，扫描结果通过 OR 与 RLO 连接。

4. 异常位 BR 存储器

—| BR |—（异常位 BR 存储器）或—| BR / |—（异常位 BR 存储器取反）触点符号用于测试状态字中 BR 位的逻辑状态。串联使用时，扫描的结果将通过 AND 与 RLO 连接；并联使用时，扫描结果通过 OR 与 RLO 连接。BR 位用于字处理向位处理的转变。

5. 数学运算结果判断

这些触点符号用于识别数学运算函数的结果与“0”的大小关系。指令扫描状态字的条件代码位 CC1 和 CC0，以确定结果与“0”的关系。串联使用时，扫描的结果将通过 AND 与 RLO 连接；并联使用时，扫描结果通过 OR 与 RLO 连接。

例如，触点符号是判断结果位不等于 0，是判断结果位取反后是否小于等于“0”等。可参见本书 4. 5 节内容。

4. 4. 4　控制指令

1. 逻辑控制指令

逻辑控制指令可以在所有逻辑块［组织块（OB）、功能块（FB）和功能（FC）］中使用，指令有无条件跳转、有条件跳转和若“否”则跳转 3 种。

跳转指令的地址用标号表示。标号最多可以包含 4 个字符。第一个字符必须是字母表中的字母，其他字符可以是字母或数字（例如，SEG3）。跳转标号指示程序将要跳转到的目标。目标标号必须位于程序段的开头。可以从梯形图浏览器中选择 LABEL，在程序段的开头输入目标标号。在显示的空框中键入标号的名称，在框中键入标签的名称。

（1）无条件跳转—(JMP)

当—(JMP) 指令左侧电源轨道与指令间没有其他梯形图元素时执行的是绝对跳转，跳转发生在块内。每个—(JMP) 指令还都必须有与之对应的目标（LABEL)。

如图 4-21 中，Network1 是一条绝对跳转指令。跳转标号是 CAS1，假设位于 Network*X*，执行跳转后，将跳过 Network1 和 Network*X* 间的指令不予执行。

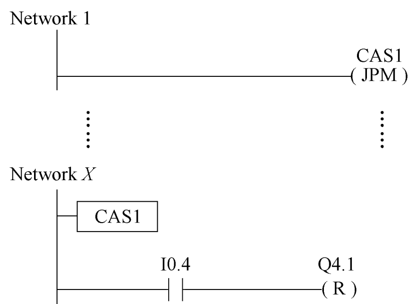


图 4-21　无条件跳转

（2）有条件跳转—(JMP)

有条件跳转和无条件跳转的区别是指令左侧电源轨道与指令间存在其他梯形图元素，则当前逻辑运算的 RLO 为“1”时执行条件跳转，否则不跳转。

（3）若“否”则跳转—(JMPN)

相当于在 RLO 为“0”时执行跳转操作，与—(JMP) 的跳转条件正好相反。

2. 程序控制指令

程序控制指令包括功能块调用及返回指令（见表 4-13）和主控继电器指令。功能块调用可以是有条件的或无条件的。

表 4-13　功能块调用及返回指令说明

指令	参数	数据类型	存储区	说明
No. —(CALL)	No.	BLOCK_FC BLOCK_SFC	—	范围取决于 CPU
DB no. BLOCK no. EN　ENO	DB no.	BLOCK_DB	I、Q、M L、D	背景数据块号
	Block no.	BLOCK_FB		功能块号
	EN	BOOL		使能端
	ENO	BOOL		使能输出
—(RET)	无	—	—	块返回

(1) 功能块调用指令

功能块调用指令—(CALL) 用于调用没有传递参数的功能 (FC) 或系统功能 (SFC)。只有 在 CALL 线圈上 RLO 为“1”时，才执行调用。

当执行—(CALL) 时，存储调用块的返回地址由当前的本地数据区代替以前的本地数据区，然后将 MA 位 (有效 MCR 位) 移位到 B 堆栈中，为被调用的功能创建一个新的本地数据区。之后，在被调用的 FC 或 SFC 中继续进行程序处理。

用 DB 指令调用功能块可以带参数。是否带参数以及带多少个参数视具体情况而不同。扫描 BR 位，可以查找 ENO。用户必须使用—(SAVE) 指令将所要求的状态分配给被调用块中的 BR 位。当调用一个功能，而被调用块的变量声明表中具有 IN、OUT 和 IN_OUT 声明时，这些变量以形式参数列表添加到调用块的程序中。当调用功能时，必须在调用位置处将实际参数分配给形式参数。功能声明中的任何初始值都没有含义。

通过声明一个数据类型为功能块的静态变量，可创建一个多重背景，只有已经声明的多重背景才会包括在程序元素目录中。多重背景的符号改变取决于是否带参数以及带多少个参数。

另外还可使用 SIMATIC 管理器中可供使用的库来选择下列块：集成在 CPU 操作系统中的块 (对于 V3 版本 STEP7 项目，为“标准库”，对于 V2 版本 STEP7 项目，为“stdlibs (V2)”)；用户自行在库中保存的块，便于多次使用。

RET 是 RETURN (返回指令) 的英文缩写，它是程序控制指令之一，用于能够有条件舍弃一个块，对于该输出，需要一个逻辑操作，RET 指令将 RLO 存储在状态字的 BR 位 (二进制结果位)。

(2) 主控继电器指令

主控继电器是一种逻辑主控开关，可以控制一段程序的执行。主控继电器指令有以下 4 种。

1) —(MCRA)：主控继电器激活。激活主控继电器功能。在该命令后，可以编程定义 MCR 区域。

2) —(MCRD)：主控继电器取消激活。在该命令后，不能编程 MCR 区域。

3) —(MCR <)：主控继电器打开。

4) —(MCR >)：主控继电器关闭。

在 MCR 堆栈中保存 RLO。MCR 嵌套堆栈为 LIFO (后入先出) 堆栈，且只能有 8 个堆栈条目 (嵌套级别)。当堆栈已满时，—(MCR <) 功能产生一个 MCR 堆栈故障 (MCRF)。

它们是输出和中间输出、置位/复位输出、RS/SR 触发器、MOVE 指令与 MCR 有关，并在打开 MCR 区域时，受保存在 MCR 堆栈中的 RLO 状态的影响。

【例】 图 4-22 所示是主控继电器指令的使用实例。有两个 MCR 区域。按如下执行该功能。

I0.0 =“1”（区域 1 的 MCR 打开）：将 I0.4 的逻辑状态分配给 Q4.1。

I0.0 =“0”（区域 1 的 MCR 关闭）：无论输入 I0.4 的逻辑状态如何，Q4.1 都为“0”。

I0.1 =“1”（区域 2 的 MCR 打开）：当 I0.3 为“1”时，将 Q4.0 设置成“1”。

I0.1 =“0”（区域 2 的 MCR 关闭）：无论 I0.3 的逻辑状态如何，Q4.0 都保持不变。

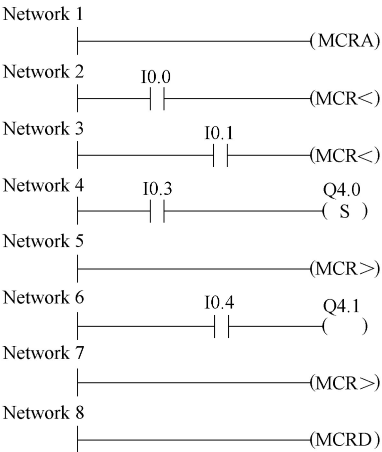


图 4-22　主控继电器指令的使用

使用主控继电器指令应注意：

取消激活 MCR 时，在 MCR 和 MCR 之间的程序段中的所有赋值都写入数值 0。这对包含赋值的所有框都有效，包括传递到块的参数在内。

当 MCR < 指令之前的 RLO = 0 时，取消激活 MCR。在某些形式参数访问或参数传递时，编译器还对在 VAR_TEMP 中定义的临时变量之后的局部数据进行写访问，以计算地址。这将把 PLC 设置成 STOP 状态，或导致未定义的运行特征。对这种情况，应释放上述命令，使其与 MCR 不相关。其实用块调用或程序跳转的方法代替主控继电器指令更容易理解。

4.5　运算指令

4.5.1　数学运算指令

1. 整数数学运算指令

整数数学运算指令对两个整数（16 和 32 位）执行加、减、乘、除等运算。整数运算指令的执行会影响状态字中的以下位：CC1 和 CC0、OV 和 OS。表 4-14 列出了运算结果对这些状态位的影响情况。

表 4-14　整数数学运算指令对状态的影响

结果的有效范围	CC1	CC0	OV	OS
0	0	0	0	*
16 位：−32 768≤结果 <0（负数） 32 位：−2 147 483 648≤结果 <0（负数）	0	1	0	*
16 位：32 767≥结果 >0（正数） 32 位：2 147 483 647≥结果 >0（正数）	1	0	0	*
结果的无效范围	**A1**	**A0**	**OV**	**OS**
下溢（加）：16 位：结果 = −65 536 32 位：结果 = −4 294 967 296	0	0	1	1

（续）

结果的无效范围	A1	A0	OV	OS
下溢（乘）：16 位：结果 < −32 768（负数） 32 位：结果 < −2 147 483 648（负数）	0	1	1	1
溢出（加，减）：16 位：结果 >32 767（正数） 32 位：结果 >2 147 483 647（正数）	0	1	1	1
溢出（乘，除）：16 位：结果 >32 767（正数） 32 位：结果 >2 147 483 647（正数）	1	0	1	1
下溢（加，减）：16 位：结果 < −32 768（负数） 32 位：结果 < −2 147 483 648（负数）	1	0	1	1
0 作除数	1	1	1	1

操作	A1	A0	OV	OS
+D：结果 −4 294 967 296	0	0	1	1
/D 或 MOD：除以 0	1	1	1	1

整数数学运算指令共有 9 条，分别是整数加、减、乘、除指令，长整数加、减、乘、除和求余指令。这些指令的梯形图形式非常相似，除了助记符，其他都相同。下面只分析整数加法指令。

在启用输入端（EN）通过一个逻辑“1”来激活 ADD_I 指令。IN1 和 IN2 相加，结果可通过 OUT 查看。如果该结果超出了整数（16 位）允许的范围，OV 位和 OS 位将为“1”并且 ENO 为逻辑“0”，这样便不执行此指令后由 ENO 连接的其他函数。

【例】 图 4-23 中，如果 I0.0 =“1”，则激活 ADD_I 指令。MW0 + MW2 相加的结果输出到 MW10。

如果结果超出整数的允许范围，ENO =“0”，取反后设置输出 Q4.0。整数加（整数加法指令说明见表 4-15。）、减、乘、除指令，长整数加、减、乘、除和求余指令的助记符分别是 ADD_I、SUB_I、MUL_I、DIV_I、ADD_DI、SUB_DI、MUL_DI、DIV_DI 和 MOD_DI。

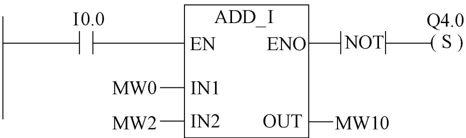


图 4-23 整数加法指令

表 4-15 整数加法指令说明

指令	参数	数据类型	存储区	说明
ADD_1 EN ENO IN1 IN2 OUT	EN	BOOL	I、Q、M、L、D	使能端
	ENO	BOOL	I、Q、M、L、D	使能输出
	IN1	INT	I、Q、M、L、D、常数	被加数
	IN2	INT	I、Q、M、L、D、常数	加数
	OUT	INT	I、Q、M、L、D	结果

2. 浮点数运算指令

这里所说的浮点数是指 32 位 IEEE 浮点数，属于 REAL 数据类型。浮点数运算分基本指令和扩展指令两类。基本指令有实数加、减、乘、除和取绝对值 5 条指令，扩展指令有浮点数求平方和平方根、浮点数求自然对数、浮点数指数运算，还有浮点数的三角函数运算，包括正弦和反正弦、余弦和反余弦、正切和反正切。

基本指令的助记符分别为：ADD_R、SUB_R、MUL_R、DIV_R 和 ABS。

扩展指令的助记符分别为：SQR 和 SQRT、LN、EXP、SIN 和 ASIN、COS 和 ACOS、TAN 和 ATAN。其中对数指令 LN 是求浮点数的自然对数；指数运算 EXP 是求浮点数的以 e（=2.718 281 83）为底的指数值。在三角函数运算中，浮点数代表一个以弧度为单位的角度，而在反三角函数运算中，求一个定义在 $-1 \leqslant$ 输入值 $\leqslant 1$ 范围内的浮点数的反三角函数值，结果是一个以弧度为单位的角度。对反正弦函数和反正切函数，结果在 $-\pi/2 \sim \pi/2$ 之间；对反余弦函数，结果在 $0 \sim \pi$ 之间。

浮点数运算指令的语法和使用比较简单，和前面讲的整数运算指令相似，只是数据类型由整数换成了浮点数。此处不再细述。

4.5.2　移位与循环指令

存储器中的一个二进制数向左移 n 位相当于乘以 2 的 n 次幂；向右移 n 位相当于除以 2 的 n 次幂。移位和循环移位指令就是完成这种操作。使用移位指令可逐位向左或向右移动输入端 IN 的内容。

由移位指令移空的位会用 0 或符号位的信号状态补上。最后移动的位的信号状态会被载入状态字的 CC1 位中。状态字的 CC0 位和 OV 位会被复位为 0。可以使用跳转指令来检测 CC1 位。

1. 移位指令

移位指令包括整数右移（SHR_I）、长整数右移（SHR_DI）、字左移和右移（SHL_W 和 SHR_W）、双字左移和右移（SHL_DW 和 SHR_DW）6 条指令。

整数右移指令用于将输入 IN 的 0～15 位逐位向右移动。16～31 位不受影响。输入 N 用于指定移位的位数。如果 N 大于 16，指令将按照 N 等于 16 的情况执行。自左移入的、用于填补空出位的位置将被赋予位 15 的逻辑状态（整数的符号位）。即当该整数为正时，这些位将被赋值“0”，而当该整数为负时，则被赋值为“1”。在输出 OUT 位置扫描移位指令的结果。如果 N 不等于 0，指令会将 CC0 位和 OV 位设为“0”。整数右移指令说明见表 4-16。

表 4-16　整数右移指令说明

指令	参数	数据类型	存储区	说明
SHR_I EN　ENO IN　OUT N	EN	BOOL	I、Q、M、L、D	使能端
	ENO	BOOL	I、Q、M、L、D	使能输出
	IN	INT	I、Q、M、L、D	输入数值
	N	WORD	I、Q、M、L、D	移位位数
	OUT	INT	I、Q、M、L、D	移位后的结果

ENO 与 EN 具有相同的信号状态。

长整数右移指令除了数据位数其余与整数右移指令相同。

SHL_W（字左移）指令通过使能（EN）输入位置上的逻辑“1”来激活。SHL_W 指令用于将输入 IN 的 0～15 位逐位向左移动。16～31 位不受影响。输入 N 用于指定移位的位数。若 N 大于 16，此命令会在输出 OUT 位置上写入“0”，并将状态字中的 CC0 位和 OV 位设置为“0”，指令将自右移入 N 个 0，用以补上空出的位。可在输出 OUT 位置扫描移位指令的结果。如果 N 不等于 0，则 SHL_W 会将 CC0 位和 OV 位设为“0”。

字和双字没有符号位，所以在移出的位都填“0”。

其他几条指令的执行过程与此相同。

2. 循环移位指令

循环移位和不循环移位的区别在于，循环移位把操作数的最高位和最低位连接起来，参与数据的移动。比如循环右移时，最低位移出的数据位移动到最高位，其余的各个位依次右移 1 位。循环左移则与此相反。循环移位通过状态字的 CC1 位进行，而 CC0 位被复位为 0。

循环移位指令只有循环左移双字（ROL_DW）和循环右移双字（ROR_DW）两条指令。输入 N 用于指定循环移位的位数。如果 N 大于 32，则双字 IN 将被循环移位（（N－1）对 32 求模，所得的余数）+1 位。如果 N 不等于 0，则指令会将 CC0 位和 OV 位设为“0”。

4.5.3　字逻辑运算指令

字逻辑运算指令按照布尔逻辑运算规则，按位运算字（16 位）和双字（32 位）对应位。如果输出 OUT 的结果不等于 0，将把状态字的 CC1 位设置为“1”。如果输出 OUT 的结果等于 0，将把状态字的 CC1 位设置为“0”。字逻辑运算指令有单字“与”（WAND_W）、“或”（WOR_W）、异“或”（WXOR_W）及双字“与”（WAND_DW）、“或”（WOR_DW）、异“或”（WXOR_DW）运算。单字逻辑与指令说明见表 4-17。

表 4-17　单字逻辑与指令说明

指令	参数	数据类型	存储区	说明
WAND_W EN　ENO IN1　OUT IN2	EN	BOOL	I、Q、M、L、D	使能端
	ENO	BOOL	I、Q、M、L、D	使能输出
	IN1	WORD	I、Q、M、L、D	逻辑运算的第一个数
	IN2	WORD	I、Q、M、L、D	逻辑运算的第二个数
	OUT	WORD	I、Q、M、L、D	结果

在使能端（EN）输入的信号状态为“1”时激活字与运算，并逐位对 IN1 和 IN2 处的两个字值进行“与”运算，按纯位模式来解释这些值，在输出端 OUT 处扫描结果。ENO 与 EN 的逻辑状态相同。其他的指令与此类似。

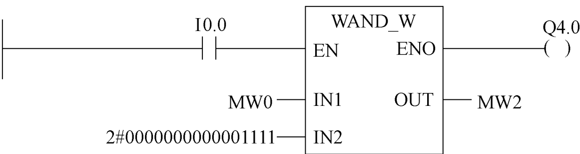


图 4-24　单字逻辑“与”指令

【例】　图 4-24 中，如果 I0.0 为“1”，则执行指令。在 MW0 的位中，只有 0～3 位是相关的，其余位被 IN2 屏蔽：

MW0＝01010101 01010101

IN2＝00000000 00001111

MW2＝00000000 00000101

如果执行了指令，则 Q4.0 为“1”。

4.6　编程举例

在 PLC 中有一些基本程序，如果我们掌握了这些基本程序的设计原理和编程技巧，对

于编写一些大型的、复杂的应用程序是非常有利的，现将程序举例如下。

4.6.1　自保持（自锁）程序

自保持（自锁）程序常用于无机械锁定开关控制中，比如照明系统的开、停，电动机的起、停等。程序如图 4-25 所示。

当 I0.0 闭合后，Q0.0 的线圈通电，随之 Q0.0 触点闭合，此后即使 I0.0 断开，Q0.0 线圈仍然保持通电，只有当动断触点 I0.1 断开时，Q0.0 线圈才断电，Q0.0 触点断开。只有重新闭合 I0.0，才能启动 Q0.0。

4.6.2　互锁程序

互锁程序用于不允许同时动作的两个继电器的控制，比如电动机的正、反转控制。互锁程序如图 4-26 所示。

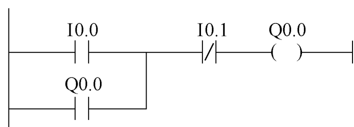


图 4-25　自保持（自锁）程序

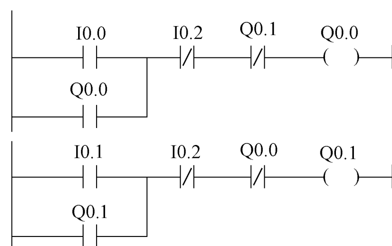


图 4-26　互锁程序

当线圈 Q0.0 先通电后，动断触点 Q0.0 断开，此时线圈 Q0.1 是不可能通电的。当线圈 Q0.1 先通电后，动断触点 Q0.1 断开，此时线圈 Q0.0 也是不可能通电的。即线圈 Q0.0、Q0.1 互相锁住，不可能同时通电，即电动机不可能同时既反转又正转。

4.6.3　延时程序

在图 4-27 中，当 I0.0 的动合触点接通后，T6 开始定时，3s 后 T6 的动合触点接通，使 Q0.0 的线圈通电。

在图 4-28 中，当 I0.0 的动合触点由断开变为接通（RLO 的上升沿）时，T7 的输出变为“1”，其动合触点闭合。在 I0.0 的下降沿，定时器开始定时，4s 后 T7 的时间值变为“0”，其动合触点断开。

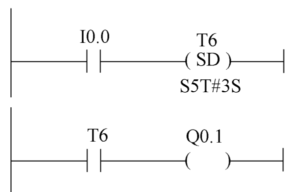


图 4-27　延时接通程序

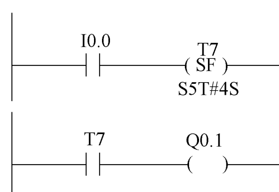


图 4-28　延时断开程序

如图 4-29 所示是利用两个定时器组合实现的长延时程序（延时 30s）。用 T2 的动合触点作为定时器 T3 的定时启动信号，当 I0.0 的动合触点接通 30s（20s + 10s）后，Q0.1 的

线圈通电。

PLC 中的普通定时器的工作与扫描工作方式有关，其定时精度受到不断变化的循环周期的影响。要获得精度较高的延时需要使用延时中断，延时中断以 ms 为单位定时。

4.6.4 分支程序

分支程序主要用于一个控制电路导致几个输出的情况。例如，开动吊车的同时打开警示灯，程序如图 4-30 所示。

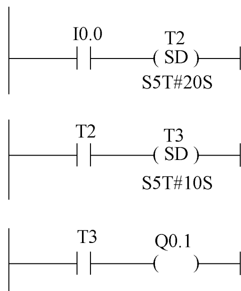


图 4-29 两个定时器组合实现长延时

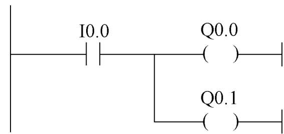


图 4-30 分支程序

当 I0.0 闭合后，线圈 Q0.0、Q0.1 同时通电。

第 5 章

S7-300/400的工程程序设计

5.1 功能与功能块的应用

5.1.1 S7-300 的用户程序结构

通过第 4 章的简单介绍我们知道，PLC 的内部程序分为系统程序和用户程序两种，在本章中，我们将详细讲解用户程序结构及构成。

用户程序是为了完成特定的自动化任务，由用户自己在 STEP 7 中编写的程序，然后下载到 CPU 中。用户程序可以完成这些工作：暖启动和热启动的初始化工作，处理过程数据（数字信号、模拟信号），对中断的响应，对异常和错误的处理。

在 STEP 7 软件中，提供了 3 种设计用户程序的方法，即线性化编程、模块化编程和结构化编程，如图 5-1 所示。

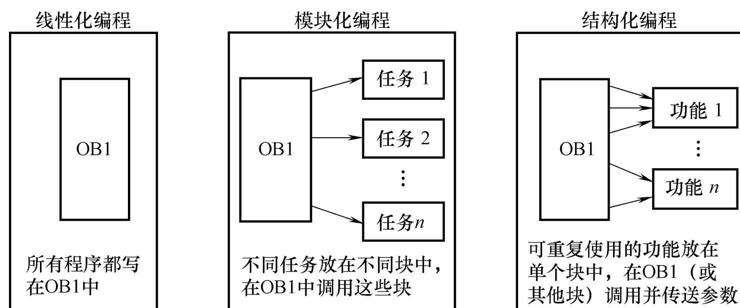


图 5-1 S7 的 3 种设计程序方法

1. 线性化编程

线性化编程是将整个用户程序放在循环控制组织块 OB1（主程序）中，处理器线性地或顺序地扫描程序的每条指令。这种方法是 PLC 最初所模拟的硬连线继电器梯形逻辑图模式，这种方式的程序结构简单，不涉及功能块（FB）、功能（FC）、数据块（DB）、局部变量和中断等比较复杂的概念，容易入门。对于许多初学者来说，建议大家在此编写简单的程序。

由于所有的指令都在一个块中，即使程序中的某些部分在大多数时候并不需要执行，但每个扫描周期都要执行所有的指令，所以没有有效地利用 CPU。此外，如果要求多次执行相同或类似的操作，需要重复编写程序。

2. 模块化编程

模块化编程是将用户程序分别写在一些块中，通常这些块都是不含参数的 FB 或 FC，每个块中包含完成一部分任务的程序，然后在主程序循环组织块 OB1 中按照顺序调用这些 FB 或 FC。模块化编程的程序被划分为若干块，易于几个人同时对一个项目编程。由于只是在需要时才调用有关的程序块，所以提高了 CPU 的利用效率。

3. 结构化编程

结构化编程将复杂的自动化任务分解为能够反映过程的工艺、功能或可以反复使用的小任务，将这些小任务通过用户程序编写一些具有相同控制过程，但控制参数不一致的程序段写在某个可分配参数的 FB 或 FC 中，然后在主程序循环组织块中可重复调用该程序块，且调用时可赋予不同的控制参数。

使用结构化编程的方法较前面两种编程方法先进，适合复杂的控制任务，并支持多人协同编写大型用户程序。结构化编程具有以下优点。

1）程序的可读性更好、更容易理解。

2）简化了程序的组织。

3）有利于对常用功能进行标准化，减少重复劳动。

4）由于可以分别测试各个程序块，所以查错、修改和调试都更容易。

5.1.2 功能的生成与调用

在 S7 软件中，结构化的用户程序是以块的形式出现的。块是一些独立的程序或者数据单元。在 STEP 7 中，块通常由组织块（OB）、功能块（FB）、系统功能块（SFB）、功能（FC）、系统功能（SFC）、背景数据块（DI）和共享数据块（DB）组成，如图 5-2 所示。各块均有相应的功能，如表 5-1 所示。

表 5-1 用户程序块

块名称	功 能 简 介	举例	块分类
组织块(OB)	确定用户程序的结构	OB1,OB2	逻辑块
功能块(FB)	由用户编写,完成需要功能,有存储区	FB2	
功能(FC)	由用户编写,完成需要功能,没有存储区	FC4	
系统功能(SFC)	集成在 S7 CPU 中,可以访问一些重要的系统功能,有存储区	SFC1	
系统功能块(SFB)	集成在 S7 CPU 中,可以访问一些重要的系统功能,没有存储区	SFB20	
背景数据块(DI)	用于指定 FB 或 SFB 传递参数或保存数据	DI10	数据块
共享数据块(DB)	用于存储用户数据,除分配给 FB 的数据外,还可以供给任何一个块来定义和使用	DB5	

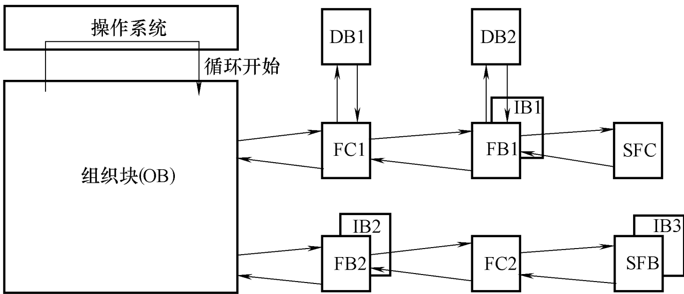


图 5-2 块结构

OB1 相当于 S7-200 系列 PLC 用户程序的主程序，除 OB1 外其他的 OB 相当于 S7-200 系列 PLC 用户程序的中断程序。FB、FC、SFB、SFC 相当于 S7-200 系列 PLC 用户程序的子程序，而 DB 和 DI 相当于 S7-200 系列 PLC 用户程序的 V 区。

在这些块中，OB、FB、FC 以及 SFC 和 SFB 都包含有由用户程序根据特定的控制任务而编写的程序代码和各程序需要的数据，因此它们称为程序块或逻辑块。DI 和 DB 不包含 STEP 7 的指令，用于存放用户数据，因此它们可统称为数据块。

5.2　组织块的应用

5.2.1　组织块与中断

组织块是 CPU 的操作系统与用户程序之间的接口。不同种类的 OB 启动的时间不同：启动 CPU 时、在循环或定时执行过程中、出错时、发生硬件触发时。

组织块都是事件触发而执行的中断程序块，按照已分配的优先级来执行，CPU 的组织块如表 5-2 所示。


注意：

并非所有的 CPU 均可处理 STEP 7 中可用的所有 OB。

表 5-2　组织块

<table>
<tr><th>OB 块分类</th><th>OB 块名称</th><th>启动时间</th><th>优先级</th><th>说明</th></tr>
<tr><td>循环执行</td><td>OB1</td><td>系统启动结束或
OB1 结束</td><td>1</td><td>自由循环</td></tr>
<tr><td rowspan="8">时间中断</td><td>OB10</td><td>时间中断 0</td><td rowspan="8">2</td><td rowspan="8">没有默认时间，
使用时需要指定时间</td></tr>
<tr><td>OB11</td><td>时间中断 1</td></tr>
<tr><td>OB12</td><td>时间中断 2</td></tr>
<tr><td>OB13</td><td>时间中断 3</td></tr>
<tr><td>OB14</td><td>时间中断 4</td></tr>
<tr><td>OB15</td><td>时间中断 5</td></tr>
<tr><td>OB16</td><td>时间中断 6</td></tr>
<tr><td>OB17</td><td>时间中断 7</td></tr>
<tr><td rowspan="4">延时中断</td><td>OB20</td><td>延时中断 0</td><td>3</td><td rowspan="4">没有默认时间，
使用时需要指定时间</td></tr>
<tr><td>OB21</td><td>延时中断 1</td><td>4</td></tr>
<tr><td>OB22</td><td>延时中断 2</td><td>5</td></tr>
<tr><td>OB23</td><td>延时中断 3</td><td>6</td></tr>
<tr><td>循环中断</td><td>OB30</td><td>循环中断 0</td><td>7</td><td>默认时间 5s</td></tr>
<tr><td>背景循环</td><td>OB90</td><td>设置最少扫描时间比
实际扫描时间大时</td><td>29②</td><td>背景循环</td></tr>
<tr><td rowspan="3">系统启动完成中断</td><td>OB100</td><td>暖启动</td><td rowspan="3">27①</td><td rowspan="3">当系统启动完毕,按照相应的
启动方式执行相应的启动 OB</td></tr>
<tr><td>OB101</td><td>热启动</td></tr>
<tr><td>OB102</td><td>冷启动</td></tr>
<tr><td rowspan="3">循环中断</td><td>OB31</td><td>循环中断 1</td><td>8</td><td>默认时间 2s</td></tr>
<tr><td>OB32</td><td>循环中断 2</td><td>9</td><td>默认时间 1s</td></tr>
<tr><td>OB33</td><td>循环中断 3</td><td>10</td><td>默认时间 500ms</td></tr>
</table>

（续）

OB 块分类	OB 块名称	启动时间	优先级	说明
循环中断	OB34	循环中断 4	11	默认时间 200ms
	OB35	循环中断 5	12	默认时间 100ms
	OB36	循环中断 6	13	默认时间 50ms
	OB37	循环中断 7	14	默认时间 20ms
	OB38	循环中断 8	15	默认时间 10ms
硬件中断	OB40	硬件中断 0	16	由模块信号触发
	OB41	硬件中断 1	17	
	OB42	硬件中断 2	18	
	OB43	硬件中断 3	19	
	OB44	硬件中断 4	20	
	OB45	硬件中断 5	21	
	OB46	硬件中断 6	22	
	OB47	硬件中断 7	23	
状态中断	OB55	状态中断	2	DPV1 中断
更新中断	OB56	更新中断		
制造商制定中断	OB57	制造商制定中断		DPV1 的 CPU 可用
SFC35 调用中断	OB60	SFC35 调用中断	25	多处理器中断
同步循环中断	OB61	同步循环中断 1		同步循环中断
	OB62	同步循环中断 2		
	OB63	同步循环中断 3		
	OB64	同步循环中断 4		
	OB65	技术同步中断		技术同步中断，仅适用于 Technology CPU
异步错误（硬件或系统错误）	OB70	I/O 冗余故障		只对于 H CPU
	OB72	CPU 冗余故障	28	
	OB73	通信冗余故障	25	
	OB80	时间错误	26，28①	超出最大循环时间
	OB81	电源错误	25，28①	电源错误
	OB82	诊断错误		输入断线（有诊断的模块）
	OB83	模板拔/插中断		S7-400 CPU 运行时模板拔/插中断
	OB84	CPU 硬件故障		MPI 接口电平错误
	OB85	程序故障		模块映像区错误
	OB86	扩展机架、DP 主站系统或用于分布式 I/O 的站故障		扩展设备或 DP 从站错误
	OB87	通信故障		读取信息格式错误
	OB88	过程中断	28	过程中断
同步错误（用户程序错误）	OB121	编程错误	导致错误的 OB 优先级相同	编程错误
	OB122	I/O 访问错误		I/O 访问错误

① 优先级 27 和 28 在优先级启动模式中是有效的。

② 优先级 29 对应于优先级 0.29，也就是说 OB90 比任何 OB 具有更低的优先级。

表 5-2 中所谓的异步错误是与硬件或者操作系统相关联的错误，这种错误与用户程序无关。异步错误组织块具有最高的优先等级，可以立即执行。表中标注①的“26，28”的意思是当前 OB 的优先等级为 26 级以下时，异步错误组织块的优先等级为 26；当前 OB 的优先等级为 26 或 27 时，异步错误组织块的优先等级为 28。表中标注①的“25，28”的意思是当前 OB 的优先等级为 25 级以下时，异步错误组织块的优先等级为 25；当前 OB 的优先等级为 25、26 或 27 时，异步错误组织块的优先等级为 28。

表 5-2 中所谓的同步错误是在执行用户程序的过程中，在用户程序的某一个特定位置上发生。当发生这种用户程序错误引起的事件时，系统会调用同步错误组织块。同步错误组织块总是与当前 OB 有相同的优先等级，所以当触发了同步错误组织块总是会被执行。当事件触发了中断事件，系统会调用相应的 OB，如果该 OB 没有被下载到 CPU 中，则 CPU 会转换到 STOP 模式；当事件触发了中断事件，系统会调用相应的 OB，如果该 OB 已经被下载到 CPU 中，尽管该 OB 是空的，CPU 还是保持运行模式。每个 OB 块系统都定义了地址为 L0.0～L19.7 的临时变量，用户可以读取这些信息。但是需要注意，必须在相应的 OB 块上编写程序而且该 OB 块被中断事件触发了才能读取这些信息。

S7　CPU 的操作系统定期执行 OB1。当操作系统完成启动后，将启动循环执行 OB1。在 OB1 中可以调用其他功能（FC、SFC）和功能块（FB、SFB）。

执行 OB1 后，操作系统发送全局数据。重新启动 OB1 之前，操作系统会将过程映像输出表写入输出模块中、更新过程映像输入表以及接收 CPU 的任何全局数据。

操作系统在运行期受监视的所有 OB 块中，OB1 的优先级最低，也就是除 OB90 之外的其他所有 OB 块均可中断 OB1 的执行。

S7 专门有监视运行 OB1 的扫描时间的时间监视器，最大扫描时间的默认为 150ms。用户编程时可以使用 SFC43 “RE_TRIGR” 来重新启动时间监视。如果用户程序超出了 OB1 的最大扫描时间，则操作系统将调用 OB80（时间错误 OB），如果没有发现 OB80，则 CPU 将转为 STOP 模式。

除了监视最大扫描时间外，还可以保证最小扫描时间。操作系统将延迟启动新循环（将过程映像输出表写入输出模块中），直至达到最小扫描时间为止。

在 OB1 中系统定义了如表 5-3 所示的本地数据，其地址从 L0.0～L19.7，地址从 L20.0 以上的本地数据允许用户定义。

表 5-3　OB1 中系统定义的本地数据

符号名称	数据类型	地址	说　明
OB1_EV_CLASS	BYTE	0.0	0～3 位 =1 事件等级；4～7 位是标识，=1 表示 OB1 激活
OB1_SCAN_1	BYTE	1.0	B#16#01：完成暖重启　B#16#02：完成热重启 B#16#03：完成主循环　B#16#04：完成冷重启 B#16#05：主站-保留站切换和“停止”上一主站之后新主站 CPU 的首个 OB1 循环
OB1_PRIORITY	BYTE	2.0	优先级 1
OB1_OB_NUMBR	BYTE	3.0	OB 编号（01）
OB1_RESERVED_1	BYTE	4.0	保留
OB1_RESERVED_2	BYTE	5.0	保留
OB1_PREV_CYCLE	INT	6.0	上一次扫描的运行时间（ms）
OB1_MIN_CYCLE	INT	8.0	自上次启动后的最小周期（ms）
OB1_MAX_CYCLE	INT	10.0	从上次启动后的最大周期（ms）
OB1_DATE_TIME	DATE_AND_TIME	12.0	调用 OB 时的 DATE_AND_TIME



注意：使用 CPU315F 和 ET200S 时应如何避免出现“通信故障”消息?

使用 CPUS7315F、ET200S 以及故障安全 DI/DO 模块，那么将调用 OB35 的故障安全程序。如果已经接受所有监控时间的默认设置值，并且愿意接收“通信故障”消息。OB35 默认设置为 100ms。如果已经将 FI/O 模块的 F 监控时间设定为 100ms，因此至少每 100ms 要寻址一次 I/O 模块。但是由于每 100ms 才调用一次 OB35，因此会发生通信故障。要确保 OB35 的扫描间隔和 F 监控时间有所差别，请确保 F 监控时间大于 OB35 的扫描间隔时间。

5.2.2 时间中断组织块

时间中断组织块可以单次运行，也可定期运行：每分钟、每小时、每天、每月、每个月末。对于每月执行的时间中断 OB，只可将 1、2、…、28 日作为起始日期。

要启动时间中断，必须先设置中断，然后再将其激活。有以下 4 种可能的启动方式：

1）自动启动时间中断。一旦使用 STEP 7 设置并激活了时间中断，即自动启动时间中断。

2）使用 STEP 7 设置时间中断，然后通过调用程序中的 SFC30“ACT-TINT”来激活它。

3）通过调用 SFC28“SET_TINT”来设置时间中断，然后通过调用 SFC30“ACT_TINT”来激活它。

4）使用 SFC39 ~ SFC42 禁用或延迟和重新启用时间中断。

由于时间中断仅以指定的时间间隔发生，因此在执行用户程序期间，某些条件可能会影响 OB 的操作。表 5-4 列出了其中的一些条件，并说明了该条件对执行时间中断 OB 的影响。

表 5-4 影响时间中断的 OB 的条件

条　件	结　果
用户程序调用 SFC29(CAN_TINT)并取消时间中断	操作系统消除了时间中断的启动事件(DATE_AND_TIME)，如果需要执行 OB，必须再次设置启动事件并在再次调用 OB 之前激活它
用户程序试图激活时间中断 OB，但未将 OB 加载到 CPU 中	操作系统调用 OB85，如果 OB85 尚未编程(装载到 CPU 中)，则 CPU 将转为 STOP 模式
当同步或更正 CPU 的系统时钟时，用户提前设置了时间并跳过时间 OB 的启动事件日期或时间	操作系统调用 OB80 并对时间 OB 的编号和 OB80 中的启动事件信息进行编码，随后操作系统将运行一次时间 OB，而不管本应执行此 OB 的次数；OB80 的启动事件信息给出了第一次跳过时间 OB 的 DATE_AND_TIME
当同步或更正 CPU 的系统时钟时，推后设置了时间以使 OB 的启动事件、日期或时间得以重复	S7-400-CPU 和 CPU 318：如果在推后设置时钟之前已激活了时间 OB，则不会再次调用它；S7-300-CPU：执行时间 OB
CPU 通过暖重启或冷重启运行	由 SFC 组态的所有时间 OB 会被改回在 STEP7 中指定的组态，如果已为相应 OB 的单次启动组态了时间中断，使用 STEP7 对其进行了设置，并将其激活，则当所组态的启动时间为已过去的时间(相对于 CPU 的实时时钟)时，会在暖重启或冷重启操作系统后调用一次 OB
当发生下一时间间隔的启动事件时，仍执行时间 OB	操作系统调用 OB80，如果 OB80 没有编程，则 CPU 转为 STOP 模式；如果装载了 OB80，则会首先执行 OB80 和时间中断 OB，然后再执行请求的中断

在 OB10 ~ OB17 中系统定义了如表 5-5 所示的本地数据，其地址从 L0.0 ~ L19.7，地址从 L20.0 以上的本地数据允许用户定义。表 5-5 中的符号以 OB10 为例。

表 5-5　OB10 中系统定义的本地数据

符号名称	数据类型	地址	说　明
OB10_EV_CLASS	BYTE	0.0	0～3 位 =1 事件等级;4～7 位是标识, =1 表示 OB 激活
OB10_STRT_INFO	BYTE	1.0	B#16#11:OB10 的启动请求 (B#16#12:OB11 的启动请求) ⋮ (B#16#18:OB17 的启动请求)
OB10_PRIORITY	BYTE	2.0	分配的优先级,默认值为 2
OB10_OB_NUMBR	BYTE	3.0	OB 编号(10～17)
OB10_RESERVED_1	BYTE	4.0	保留
OB10_RESERVED_2	BYTE	5.0	保留
OB10_PERIOD_EXE	WORD	6.0	OB 以指定的时间间隔执行: W#16#0000:单次 W#16#0201:每分钟一次 W#16#0401:每小时一次 W#16#1001:每天一次 W#16#1201:每周一次 W#16#1401:每月一次 W#16#1801:每年一次 W#16#2001:月末
OB10_RESERVED_3	INT	8.0	保留
OB10_RESERVED_4	INT	10.0	保留
OB10_DATE_TIME	DATE_AND_TIME	12.0	调用 OB 时的 DATE_AND_TIME

【例】　使用 STEP 7 设置并激活时间中断。

1）首先建立一个完整项目，项目名称为“设置时间中断”，如图 5-3 所示。



图 5-3　建立一个完整项目

2）然后单击项目中的“SIMATIC　300（1）”，双击右边的硬件图标，如图 5-4 所示。



图 5-4　打开硬件组态画面

3）将自动弹出硬件组态画面，如图 5-5 所示，把电源和 CPU 等放到机架相应的位置上。在硬件组态画面，双击机架上的 CPU，如图 5-5 所示，将弹出 CPU 属性画面。

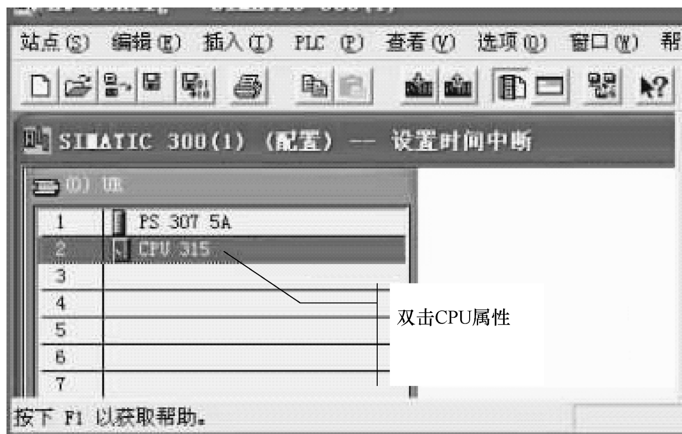


图 5-5 硬件组态画面

4）在 CPU 属性画面单击“时刻中断”，打开时间中断设置画面，如图 5-6 所示，选择“激活”，要求每分钟执行并写上开始执行 OB10 的日期和时间，图中的开始日期时间是 2007 年 3 月 28 日 21 时 21 分，然后单击“确定”。

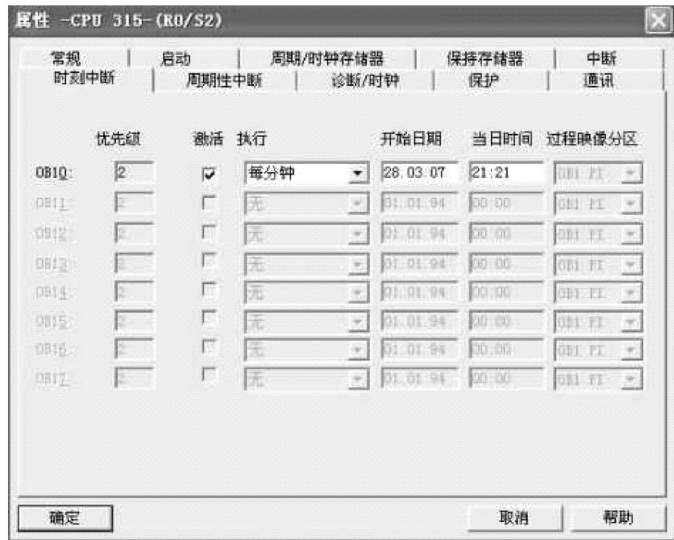


图 5-6 设置 OB10 时间中断

5）最后在硬件组态画面单击保存和编译快捷图标，如图 5-7 所示完成保存和编译。

6）在管理画面单击“块”，然后在右边块的目录下单击鼠标右键，在弹出的画面中单击“插入新对象”→“组织块”，如图 5-8 所示。

7）在生成组织块的过程中，选择组织块 OB10 及写上附加的信息，如图 5-9 所示，然后单击“确定”。

8）在管理画面的块目录里双击 OB10 的图标，如图 5-10 所示，打开 OB10 编程界面。

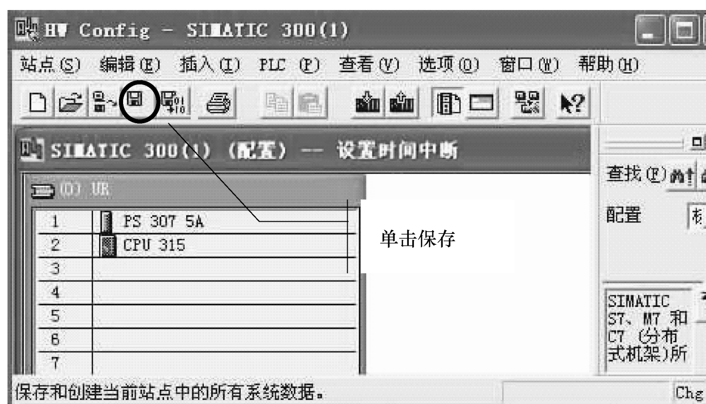


图 5-7　把硬件组态信息编译并保存



图 5-8　在管理画面插入 OB10



图 5-9　在生成 OB10 过程中写上附加信息



图 5-10 在管理画面打开 OB10

9）在 OB10 里编写如图 5-11 所示的程序，然后保存。

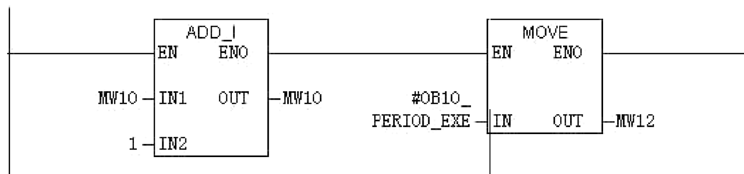


图 5-11 在 OB10 里编写的程序

10）在管理画面的块目录里单击“SIMATIC 300（1）”，然后单击下载快捷图标，如图 5-12 所示，表示把整个项目的信息下载到 CPU 中。



图 5-12 把整个项目（站）的信息下载到 CPU（或仿真 PLC）中

11）最后把 CPU 扳动到运行模式，监控 OB10 的程序状态，可以看到每分钟 MW10 的数值会加 1，如图 5-13 所示。

【例】 通过调用 SFC28“SET_TINT”来设置时间中断，然后通过调用 SFC30“ACT_TINT”来激活它。

1）首先建立一个项目并完成项目，该项目名为“设置时间中断 2”，如图 5-14 所示。然后进行硬件配置并把硬件配置的信息保存编译，如图 5-15 所示。

2）在管理画面的块目录里插入 OB10，如图 5-16 所示。

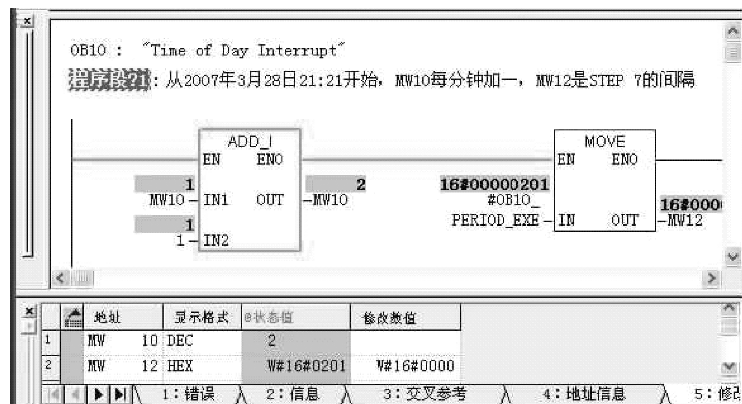


图 5-13　在 CPU 运行状态下监控 OB10 的程序状态



图 5-14　建立一个项目



图 5-15　完成硬件组态

3）在 OB10 生成的过程中，写上附加信息，如图 5-17 所示。

4）打开 OB10，出现如图 5-18 所示画面。在 OB10 里编写程序，如图 5-19 所示，然后保存。

5）在管理画面的块目录里打开 OB1，如图 5-20 所示。在 OBI 里编写程序，如图 5-21 所示，然后保存。

6）在管理画面的块目录里插入 OB80 并打开 OB80，如图 5-22 所示。在 OB80 里编写如图 5-23 所示程序并保存。

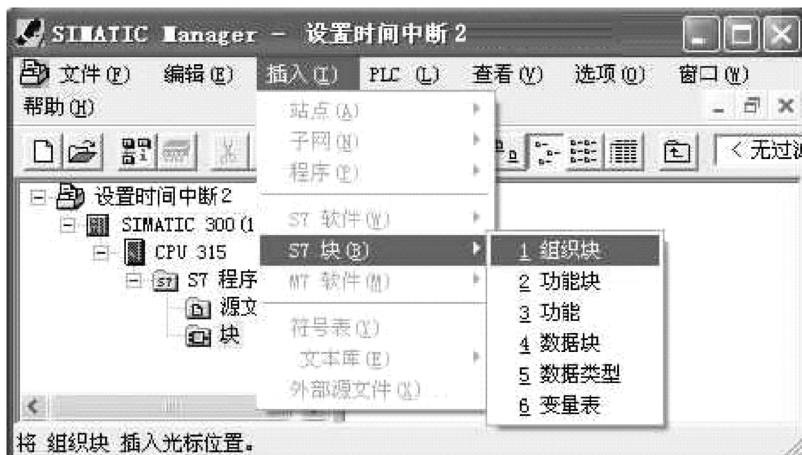


图 5-16 在管理画面插入 OB10

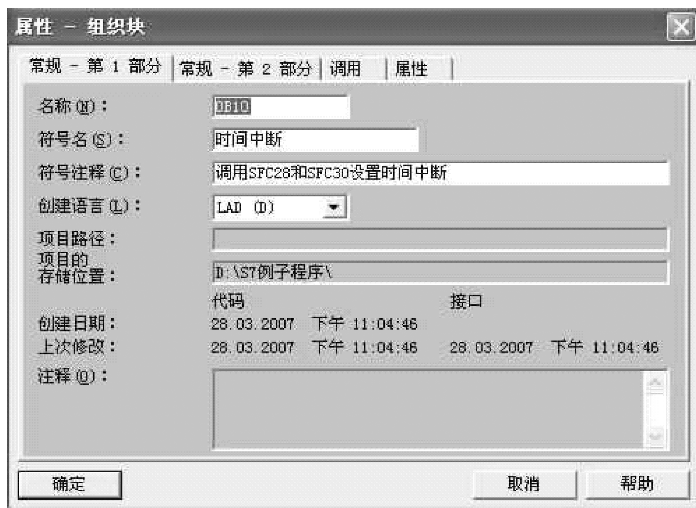


图 5-17 填写附加信息

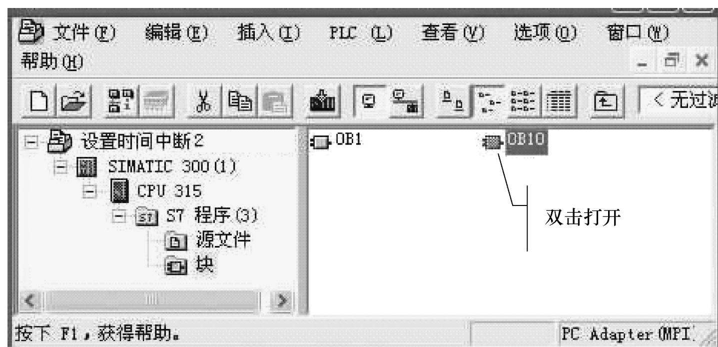


图 5-18 在管理画面打开 OB10

OB10 :　"Time of Day Interrupt"

程序段1: 要求从2007年3月28日23：10开始，
每分钟MW10加3。MW12监控设置的间隔

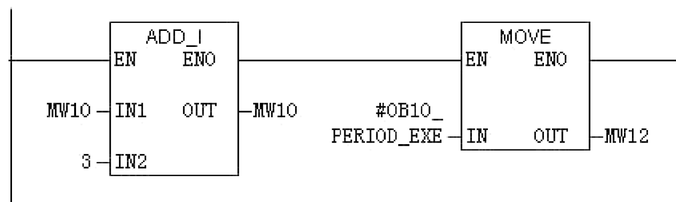


图 5-19　在 OB10 里编写的程序



图 5-20　在管理画面打开 OB1

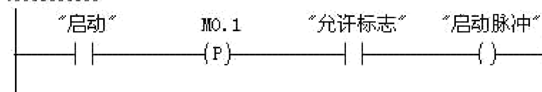
- 接口
  - TEMP
    - OB1_EV_CLASS
    - OB1_SCAN_1
    - OB1_PRIORITY
    - OB1_OB_NUMBR
    - OB1_RESERVED_1
    - OB1_RESERVED_2
    - OB1_PREV_CYCLE
    - OB1_MIN_CYCLE
    - OB1_MAX_CYCLE
    - OB1_DATE_TIME
    - RIQI_SHIJIAN

内容：'环境\接口\TEMP'

名称	数据类型
OB1_EV_CLASS	Byte
OB1_SCAN_1	Byte
OB1_PRIORITY	Byte
OB1_OB_NUMBR	Byte
OB1_RESERVED_1	Byte
OB1_RESERVED_2	Byte
OB1_PREV_CYCLE	Int
OB1_MIN_CYCLE	Int
OB1_MAX_CYCLE	Int
OB1_DATE_TIME	Date_And_Time
RIQI_SHIJIAN	Date_And_Time

OB1 :　"Main Program Sweep (Cycle)"

程序段1: 根据启动标志状态，启动OB10的时间中断。



符号信息:

启动	M0.0
允许标志	M0.3
启动脉冲	M0.2

程序段?2：合成日期和时间的值

功能FC3将数据格式DATE和TIME_OF_DAY (TOD)组合在一起，并将这些格式转换为数据类型格式DATE_AND_TIME (DT)。
输入值IN1必须在限定值DATE#1990-01-01和DATE#2089-12-31之间。(不检查此值)此功能不报告任何错误。

图 5-21　在 OB1 里编写的程序

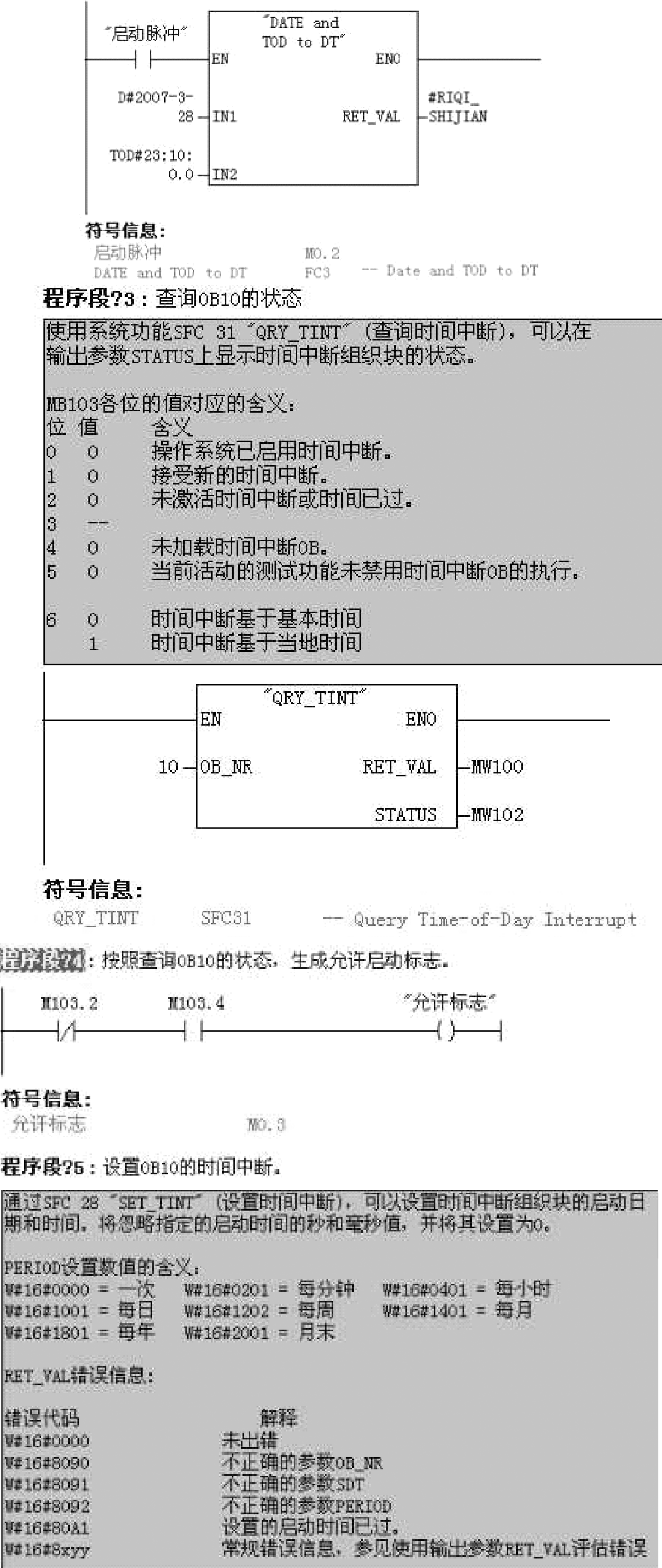
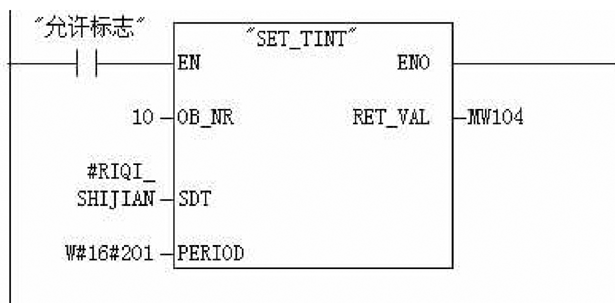


图 5-21 在 OB1 里编写的程序（续）



符号信息：

允许标志　　M0.3

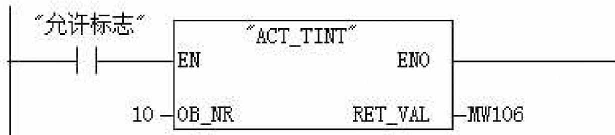
SET_TINT　　SFC28　　-- Set Time-of-Day Interrupt

程序段?6：激活OB10时间中断。

使用SFC 30 "ACT_TINT" (激活时间中断)，可以激活一个时间中断组织块。

RET_VAL错误信息

错误代码	解释
W#16#0000	未出错。
W#16#8090	不正确的参数OB_NR。
W#16#80A0	没有为相应的时间中断OB设置启动日期/时间。
W#16#80A1	激活的时间已过。只有当选择"执行 = 一次"时才会出现此错误。
W#16#8xyy	常规错误信息，参见使用输出参数RET_VAL评估错误



符号信息：

允许标志　　M0.3

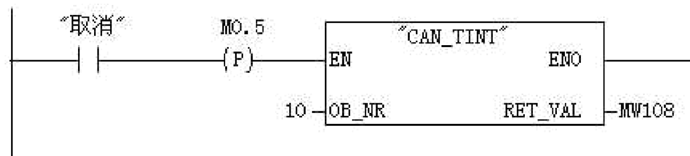
ACT_TINT　　SFC30　　-- Activate Time-of-Day Interrupt

程序段?7：取消OB10时间中断。

使用SFC 29 "CAN_TINT" (取消时间中断)，可以取消一个已激活的时间中断组织块

RET_VAL输出错误信息：

错误代码	解释
W#16#0000	未出错。
W#16#8090	不正确的参数OB_NR
W#16#80A0	没有为时间中断OB指定启动日期/时间
W#16#8xyy	常规错误信息，参见使用输出参数RET_VAL评估错误



符号信息：

取消　　M0.4

CAN_TINT　　SFC29　　-- Cancel Time-of-Day Interrupt

图 5-21　在 OB1 里编写的程序（续）

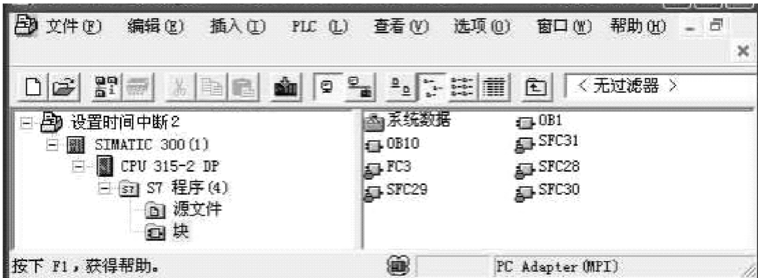


图 5-22　编写完 OB1 程序后管理画面中的块信息

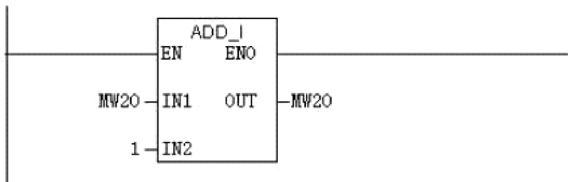


图 5-23　在 OB80 里编写的程序

7）如图 5-24 所示，是完成 OB1 和 OB80 的程序后在管理画面的块目录里的内容。然后在管理画面把整个项目的信息下载到 CPU 中（或 PLCSIM 仿真 PLC）。



图 5-24　在管理画面把整个站的信息下载到 CPU 中

8）在调试的时候重点监控 M103.2 和 M103.4 的状态。运行 CPU，可以看到 M103.4 = 1，表示 OB10 已经下载到 CPU；当接通 M0.2 时，可以看到 M103.2 = 1，表示激活了 OB10，并看到 MW10 每分钟加 1；当接通 M0.4 时，看到 M103.2 = 0，表示 OB10 已经被取消激活，MW10 停止加 1。

5.2.3　硬件中断组织块

S7 提供了 8 个独立的硬件中断，每一中断都具有自己的组织块。硬件中断组织块是对具有中断能力的数字量信号模块（SM）、通信处理器（CP）和功能模块（FM）信号变化进行中断响应。

对于具有中断能力的数字量信号模块（SM），可以使用 STEP 7 软件在硬件组态时设置硬件中断，也可以使用 SFC55 ~ SFC57 为模块的硬件中断分配参数来实现设置硬件中断。对

于具有中断能力的通信处理器（CP）和功能模块（FM），可以使用 STEP 7 软件在硬件组态时按照向导的对话框设置相应的参数来实现设置中断。对于具有中断能力的数字量信号模块（SM）使用 STEP 7 软件在硬件组态时，可选择在输入信号的上升沿或下降沿触发硬件中断。如果不激活两个复选框中的任何一个，硬件中断就不会被各自的输入信号所触发，如图 5-25 所示；如果同时激活这两个复选框，模块会在上升沿和下降沿均产生一个硬件中断，如图 5-26 所示，图中选择 0 通道的上升沿和下降沿。



图 5-25　不激活复选框



图 5-26　同时激活两个复选框

在设置数字量信号模块信号变化响应中断时，在 4 个通道组成的组中，可为每个通道设定一个输入延迟。方法是在输入域中单击鼠标左键，然后在打开的相关选择菜单中选择所需的延迟值，如图 5-27 所示，图中选择 0～3 通道延时时间是 15ms。

硬件中断被模块触发后，操作系统将标识哪一个槽的模块以及与之相应的硬件中断 OB。如果此 OB 的优先级高于当前激活的优先级，则将启动该 OB，当执行完此硬件中断 OB 后，



图 5-27　选择输入延时

将发送通道确认信号。

如果在对硬件中断进行标识和确认的这段时间内，发生了触发硬件中断的事件时：

1）如果该事件发生在先前触发硬件中断的通道中（比如同一个输入的上升沿），则新中断丢失；

2）如果该事件发生在同一模块的另一通道中，通常不会立即触发任何硬件中断，但是此中断不会丢失，而是在确认当前激活的硬件中断后被触发；

3）如果因来自另一模块中的硬件中断而使某一硬件中断被触发，并且其 OB 当前处于激活状态，则将记录新请求（相当于把请求排队）并且在 OB 空闲时对其进行处理。

可以使用 SFC39 ~ SFC42 来禁用或延迟，并重新启用硬件中断。在 OB40 ~ OB47 中系统定义了如表 5-6 所示的本地数据，其地址从 L0.0 ~ L19.7，地址从 L20.0 以上的本地数据允许用户定义。表 5-6 中的符号以 OB40 为例。

表 5-6　OB40 中系统定义的本地数据

符号名称	数据类型	地址	说明
OB40_EV_CLASS	BYTE	0.0	事件等级和标识符 B#16#11：中断处于激活状态
OB40_STRT_INF	BYTE	1.0	B#16#41：通过中断线 1 中断 B#16#42：通过中断线 2 中断(仅限 S7-400) B#16#43：通过中断线 3 中断(仅限 S7-400) B#16#44：通过中断线 4 中断(仅限 S7-400) B#16#45：WinAC，通过 PC 触发中断
OB40_PRIORITY	BYTE	2.0	分配的优先级：默认值 OB40 ~ OB47 分别是 16 ~ 23
OB40_OB_NUMBR	BYTE	3.0	OB 编号(40 ~ 47)
OB35_RESERVED_1	BYTE	4.0	保留
OB40_IO_FLAG	BYTE	5.0	输入模块：B#16#54 输出模块：B#16#55
OB40_MDL_ADDR	WORD	6.0	触发中断的模块的逻辑地址(字节为单位，是起始字节)

（续）

符号名称	数据类型	地址	说　明
OB40_POINT_ADDR	INT	8.0	对于数字模块：模块上具有输入状态的位域（位 0 对应于第一个输入）可在给定模块的说明中找到为模块中的通道分配的从 OB40_POINT_ADDR 起始的位 对于模拟模块：位域，指出哪个通道已超出哪条限制对于 CP 或 IM：模块中断状态（与用户程序无关）
OB40_DATE_TIME	DATE_AND_TIME	12.0	调用 OB 时的 DATE_AND_TIME

【例】　硬件中断组织块例子。

要求：使用 CPU 314C-2DP 集成的 I/O 设置硬件中断，当 I124.0 下降沿时而且此时 OB40 被激活，则置位 Q124.0；当 I124.1 上升沿时而且此时 OB40 被激活，则复位 Q124.0。当 I124.2 上升沿时，取消激活 OB40；当 I124.3 上升沿时，激活 OB40。

1）首先建立一个项目，并完成项目的目录配置，如图 5-28 所示。然后选中 SIMATIC 300（1），双击硬件图标打开硬件组态画面并完成硬件配置，如图 5-29 所示。

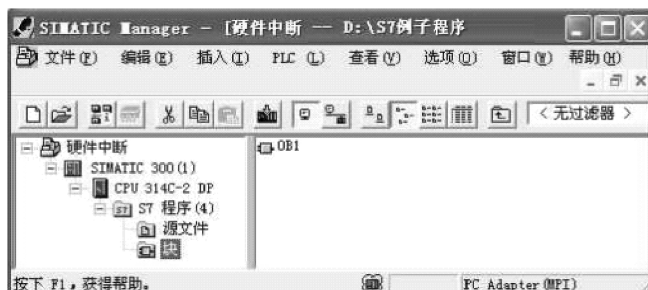


图 5-28　新建硬件中断项目



图 5-29　组态硬件配置

2）在图 5-29 中，双击机架上 CPU 集成的“DI24/DO16”，打开 DI24/DO16 属性组态画面，如图 5-30 所示，使用默认地址配置输入地址是 IB124 ~ IB126，输出地址是 QB124 ~ QB125。



图 5-30　选择中断条件

3）在图 5-30 中，选择 0 通道下降沿和 1 通道上升沿，也就是 I124.0 的下降沿和 I124.1 的上升沿触发中断事件，默认的中断组织块是 OB40，然后单击“确定”。最后编译保存，生成系统数据，如图 5-31 所示。

4）在管理画面的块目录里插入 OB40，并打开 OB40，在 OB40 中编写如图 5-32 所示的程序并保存。

5）为了能够取消激活和重新激活中断，在管理画面的块目录打开 OB1，在 OB1 中编写如图 5-33 所示程序并保存。

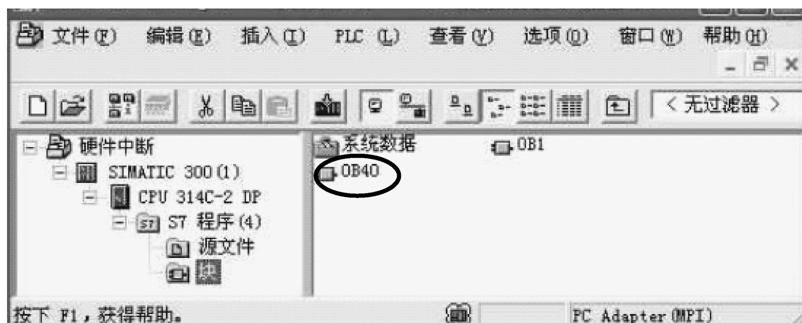


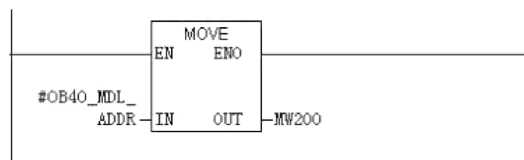
图 5-31　完成块目录里出现 OB40

6）在管理画面把整个项目（如图 5-34 所示）的信息下载到 CPU，然后运行 CPU。

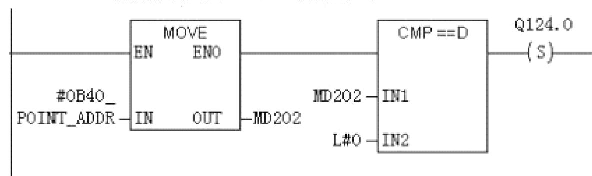
当在 I124.0 断开瞬间，可以看到 Q124.0 接通了；当在 I124.1 接通瞬间，可以看到 Q124.0 复位了。当 I124.2 出现上升沿后，使 I124.0 断开，这时 Q0.0 不会接通，说明 OB40

OB40 : "Hardware Interrupt"硬件中断OB40.

程序段?1: 读取触发中断的模块的逻辑基址，显示在MW200中。



程序段?2 : 读取模块中产生中断的通道位置，
如果是0通道（I124.0)则置位Q124.0



程序段?3 : 读取模块中产生中断的通道位置，
如果是1通道（I124.1)则复位Q124.0

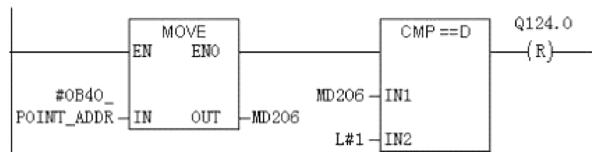


图 5-32　在 OB40 里编写的程序

被取消激活了；当 I124. 3 出现上升沿后，使 I124. 0 断开，可以看到 Q0. 0 会接通。当在 I124. 1 接通瞬间，可以看到 Q124. 0 复位了，说明 OB40 重新被激活了。

【例】 利用仿真软件，进行硬件中断组织块。

在管理画面把整个项目（见图 5-34）的信息下载到 PLCSIM 仿真 CPU。

1）首先把 PLCSIM 仿真 CPU 扳动到运行模式，然后单击“Execute”→“Trigger Error OB”→“Hardware Interrupt（OB40-OB47）”，在 PLCSIM 仿真 CPU 中打开硬件中断组织块，同时插入 IB124 和 QB124 的 I/O 面板，如图 5-36 所示。

2）在 Hardware Interrupt OB（40-47）面板上的“Module address”栏写上 124，在“Module status（POINT_ADDR）”栏写上 0，然后单击“Apply”，意思是模拟 I124. 0 产生一个脉冲信号，这时可以看到 Q124. 0 接通了，表明 OB40 已经被激活并执行，这时监看 Hardware Interrupt OB（40-47）面板上的“Interrupt OB”栏出现 40 字样，说明该中断执行 OB40；然后在“Module address”栏保留 124，在“Module status（POINT_ADDR）”栏写上 1，然后单击“Apply”，意思是模拟 I124. 1 产生一个脉冲信号，这时可以看到 Q124. 0 复位了，表明 OB40 已经被激活并执行。当在 IB124 面板中接通 I124. 2，在“Module address”栏写上 124，在“Module status（POINT_ADDR）”栏写上 0，然后单击“Apply”，这时可以看到 Q124. 0 不会接通，表明 OB40 已经被取消激活。

在 IB124 面板中接通 I124. 3，在“Module address”栏写上 124，在“Module status（POINT_ADDR）”栏写上 0，然后单击“Apply”，这时可以看到 Q124. 0 接通了，表明 OB40 已经被重新激活了。

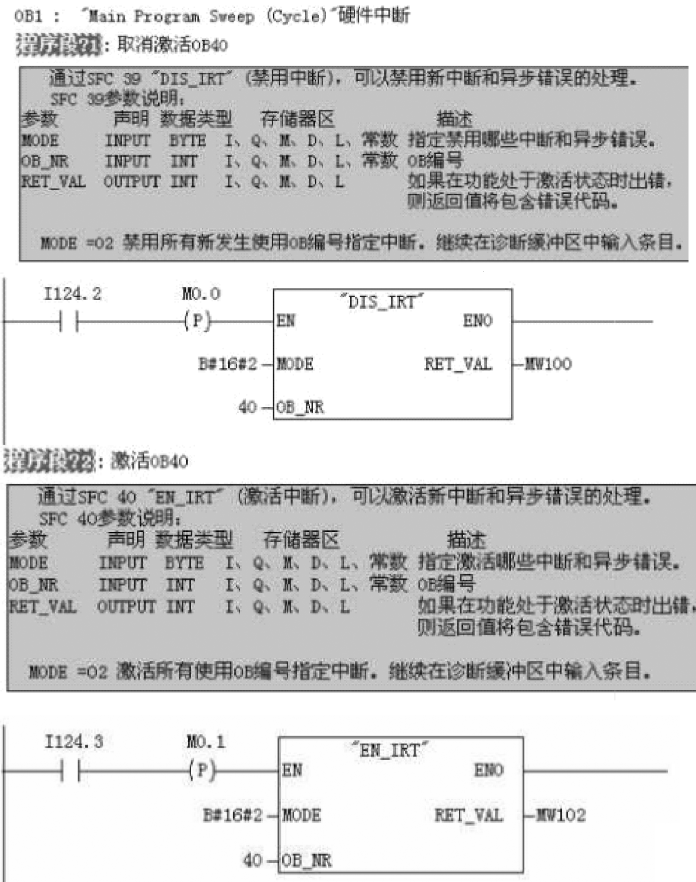


图 5-33　在 OB1 里编写的程序



图 5-34　完成 OB1 和 OB40 程序后管理界面出现的块目录

5.2.4　延时中断组织块

S7 提供了 4 个在指定延迟后执行的 OB（OB20 ~ OB23）。每个延时 OB 均可通过调用 SFC32（SRT_DINT）来启动。延迟时间是 SFC32 的一个输入参数。

当用户程序调用 SFC32（SRT_DINT）时，需要提供 OB 编号、延迟时间和用户专用的标识符。经过指定的延迟后，相应的 OB 将会启动。还可使用 SFC33 取消尚未启动的延时中

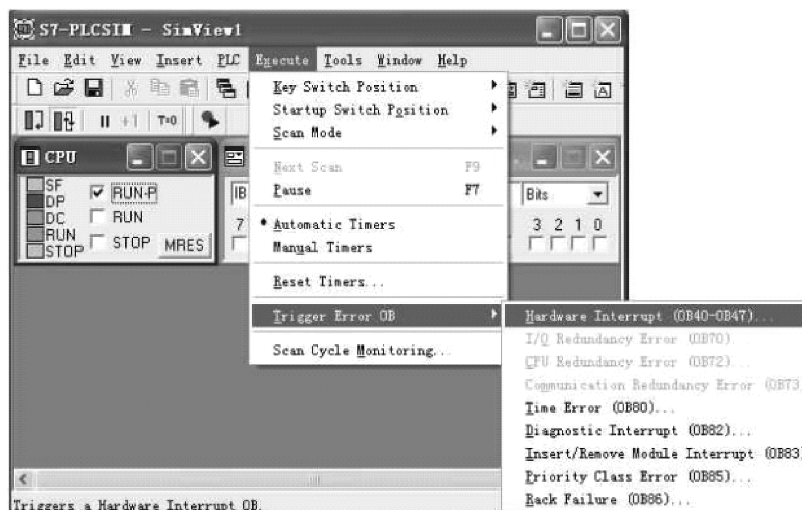


图 5-35　在 PLCSIM 仿真 CPU 里打开硬件中断组织块

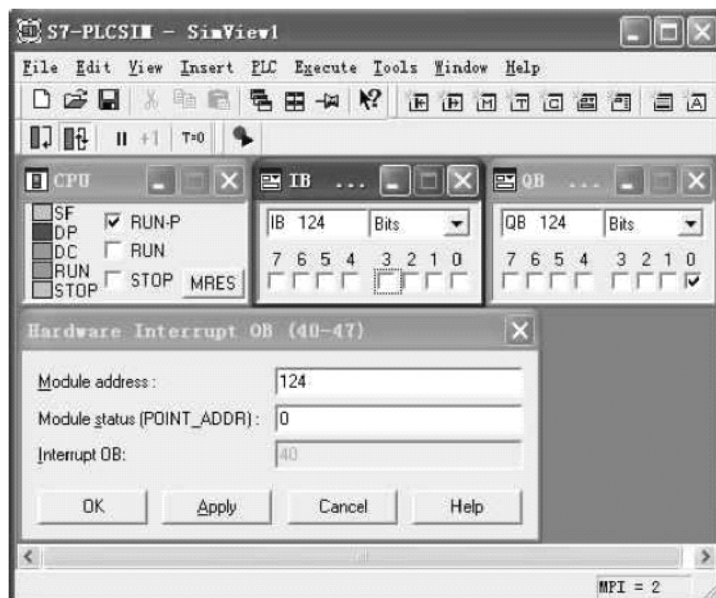


图 5-36　使用 PLCSIM 调试硬件中断画面

断，可以使用 SFC34 访问延时中断组织块的状态。可使用 SFC39 ~ SFC42 来禁用或延迟并重新使能延迟中断。只有当 CPU 处于 RUN 模式下时才会执行延时 OB。暖重启或冷重启将清除延时 OB 的所有启动事件。

延迟时间（单位为 ms）和 OB 编号一起传送给 SFC32，时间到期后，操作系统将启动相应的 OB。设置延时中断最基本的步骤是：调用 SFC32（SRT_DINT），并将延时中断 OB 作为用户程序的一部分下载到 CPU。

如果发生了操作系统试图启动一个尚未装载的 OB，并且用户在调用 SFC32“SRT_DINT”时指定了其编号，或在完全执行延时 OB 之前发生延时中断的下一个启动事件时，操

作系统将调用异步错误 OB。在 OB20 ~ OB23 中系统定义了如表 5-7（表 5-7 中的符号以 OB20 为例。）所示的本地数据，其地址从 L0.0 ~ L19.7，地址从 L20.0 以上的本地数据允许用户定义。

表 5-7　OB20 中系统定义的本地数据

符号名称	数据类型	地址	说　　明
OB20_EV_CLASS	BYTE	0.0	0 ~ 3 位 = 1 事件等级；4 ~ 7 位是标识，= 1 表示 OB 激活
OB20_STRT_INF	BYTE	1.0	B#16#21：OB20 的启动请求 （B#16#22：OB21 的启动请求） （B#16#23：OB22 的启动请求） （B#16#24：OB23 的启动请求）
OB20_PRIORITY	BYTE	2.0	分配的优先级：默认值 OB20 ~ OB23 分别是 3 ~ 6
OB20_OB_NUMBR	BYTE	3.0	OB 编号（20 ~ 23）
OB20_RESERVED_1	BYTE	4.0	保留
OB20_RESERVED_2	BYTE	5.0	保留
OB20_SIGN	WORD	6.0	用户 ID：来自调用 SFC32（SRT_DINT）的输入参数 SIGN
OB20_DTIME	TIME	8.0	已组态的延迟时间（单位为 ms）
OB20_DATE_TIME	DATE_AND_TIME	12.0	调用 OB 时的 DATE_AND_TIME

【例】　通过调用 SFC32 来设置延时中断，通过调用 SFC33 来取消延时中断。

1）首先建立完整的项目目录，如图 5-37 所示，再完成硬件组态并编译保存，如图5-38 所示。

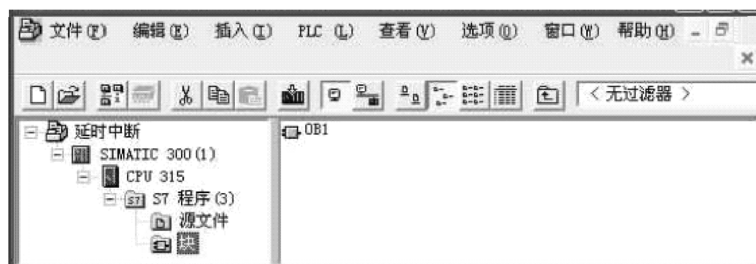


图 5-37　建立完整的项目目录

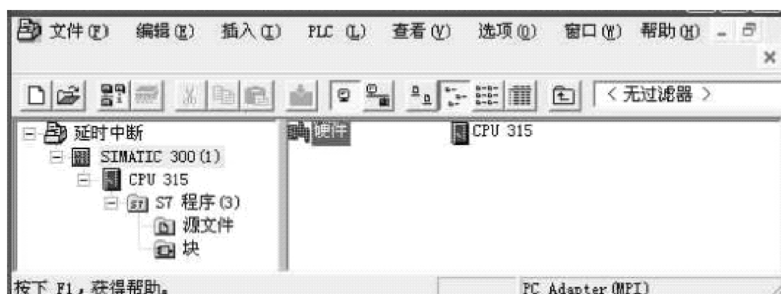


图 5-38　完成的硬件组态

2）在管理画面的块目录里打开 OB1，如图 5-39 所示。在 OB1 里编写如图 5-40 所示的程序并保存。

3）在管理画面的块目录里插入 OB20，在 OB20 里编写如图 5-41 所示程序并保存；在管理画面的块目录里插入 OB85，在 OB85 里编写如图 5-42 所示程序并保存。



图 5-39　在管理画面打开 OB1

OB1 : "Main Program Sweep (Cycle)" 设置延时中断
注释：当M0.1上升沿时，设置并激活延时中断OB20

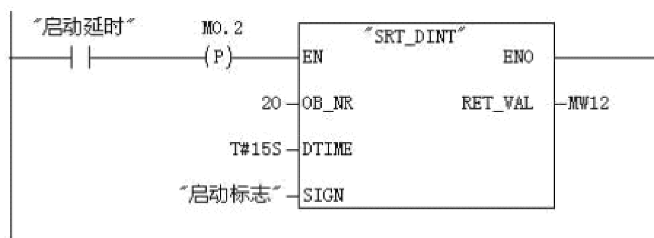
通过SFC 32 "SRT_DINT" (启动延时中断)，可以在延迟时间过去(参数DTIME)后立即启动调用延时中断组织块的延时中断。
使用SIGN参数，可以输入用于标识延时中断开始的标识符。执行指定的OB时，DTIME和SIGN值将再次显示在该OB的启动事件信息中。

SFC 32 参数说明：

参数	声明	数据类型	存储器区	描述
OB_NR	INPUT	INT	I、Q、M、D、L、常数	将在延时后启动的OB的编号。
DTIME	INPUT	TIME	I、Q、M、D、L、常数	延时值(1到60000毫秒)。
SIGN	INPUT	WORD	I、Q、M、D、L、常数	调用延时中断OB时将显示在启动事件信息中的标识符。
RET_VAL	OUTPU	TINT	I、Q、M、D、L	如果在系统功能处于激活状态时出错，则RET_VAL的实际参数将包含错误代码。

RET_VAL错误信息

错误代码	解释
W#16#0000	未出错。
W#16#8090	不正确的参数OB_NR
W#16#8091	不正确的参数DTIME



符号信息：

启动延时	M0.1	
SRT_DINT	SFC32	-- Start Time-Delay Interrupt
启动标志	MW10	

图 5-40　在 OB1 里编写的程序

程序段?2：查询OB20的状态

通过SFC 34 "QRY_DINT"（查询延时中断），可以查询延时中断OB的状态。延时中断由组织块OB20到OB23管理。

SFC 34 参数说明：

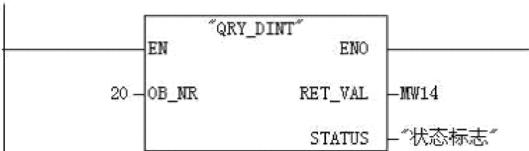
参数	声明	数据类型	存储器区	描述
OB_NR	INPU	TINT	I、Q、M、D、L、常数	将查询其STATUS的OB的编号。
RET_VAL	OUTPU	TINT	I、Q、M、D、L	如果在处理功能期间出错，则包含错误代码。
STATUS	OUTPUT	WORD	I、Q、M、D	

输出参数STATUS的含义：

位	值	含义
0	0	操作系统已启用延时中断。
1	0	未拒绝新的延时中断。
2	0	未激活延时中断或时间已过。
3	--	
4	0	未加载延时中断OB。
5	0	当前活动的测试功能未禁用延时中断OB的执行。

RET_VAL错误信息：

错误代码	解释



符号信息：

QRY_DINT　SFC34　-- Query Time-Delay Interrupt
状态标志　MW16

程序段?3：当M0.3上升沿时，取消OB20延时中断。

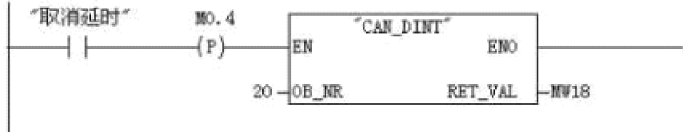
通过SFC 33 "CAN_DINT"（取消延时中断），可以取消已启动的延时中断（参见使用SFC 32 "SRT_DINT"启动延时中断）。这样，将不调用延时中断OB。

SFC 33 参数说明：

参数	声明	数据类型	存储器区	描述
OB_NR	INPUT	INT	I、Q、M、D、L、常数	将取消的OB的编号（OB20到OB23）。
RET_VAL	OUTPUT	INT	I、Q、M、D、L	如果在功能处于激活状态时出错，则将包含错误代码。

RET_VAL错误信息：

错误代码	解释
W#16#0000	未出错。
W#16#8090	不正确的参数OB_NR
W#16#80A0	未启动延时中断。
W#16#8xyy	常规错误信息，参见使用输出参数RET_VAL评估错误



符号信息：

取消延时　M0.3
CAN_DINT　SFC33　-- Cancel Time-Delay Interrupt

程序段?4：当OB20没有被激活时，M0.5=1



图 5-40　在 OB1 里编写的程序（续）

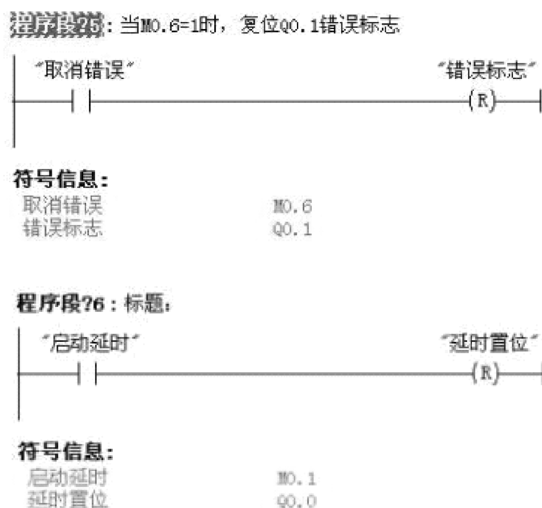


图 5-40　在 OB1 里编写的程序（续）

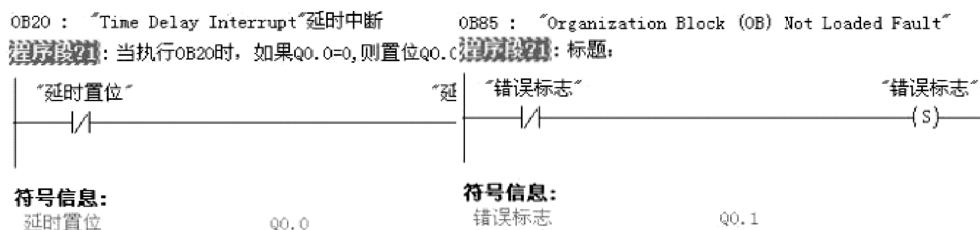


图 5-41　在 OB20 里编写的程序　　　　图 5-42　在 OB85 里编写的程序

4）完成 OB1、OB20 和 OB85 后管理画面的块目录如图 5-43 所示。



图 5-43　完成的 OB1、OB20 和 OB85 后管理界面里的块目录

5）最后在管理画面把延时中断的整个项目下载到 CPU（也可以是 PLCSIM 仿真 PLC），调试时注意监控 M17.2 和 M17.4 的状态。当把 CPU 扳动到运行模式时，可以看到 M17.4 = 1，说明 OB20 已经下载到 CPU 中。当接通 M0.1 启动延时中断时，M17.2 = 1；当延时时间到，Q0.0 = 1。当接通 M0.3 取消延时中断或延时时间到后，M17.2 = 0。

5.2.5　循环中断组织块

S7 提供了 9 个循环中断 OB（OB30 ~ OB38），可以指定的固定时间间隔来中断用户程

序。循环中断 OB 的等距启动时间是由时间间隔和相位偏移量决定的。

用户编写程序时，必须确保每个循环中断 OB 的运行时间远远小于其时间间隔。如果因时间间隔已到期，在预期的再次执行前未完全执行循环中断 OB，则启动时间错误 OB（OB80），稍后将执行导致错误的循环中断。

在编写程序时如果有多个循环中断 OB，设置要求循环中断的时间间隔又成整数倍，那么有可能会出现处理循环中断的时间过长而引起超出扫描周期时间错误。为了避免这种情况，最好定义一个偏移量时间，偏移量时间务必要小于间隔时间。偏移量时间使循环间隔时间已到，延时偏移量的时间再执行循环中断，偏移量时间不会影响循环中断的周期。用户编写程序时可使用 SFC39 ~ SFC42 来禁用或延迟，并重新启用循环中断。使用 SFC39 来取消激活循环中断，使用 SFC40 来激活循环中断。

在 OB30 ~ OB38 中系统定义了如表 5-8 所示的本地数据（表 5-8 中的符号以 OB35 为例），其地址从 L0.0 ~ L19.7，地址从 L20.0 以上的本地数据允许用户定义。

表 5-8　OB35 中系统定义的本地数据

符号名称	数据类型	地址	说明
OB35_EV_CLASS	BYTE	0.0	事件等级和标识符 B#16#11：中断处于激活状态
OB35_STRT_INF	BYTE	1.0	B#16#30：具有特殊标准的循环中断 OB 的启动请求（仅适用于 H-CPU，且仅当对其进行了明确组态后） B#16#31：OB30 的启动请求 ⋮ B#16#39：OB38 的启动请求
OB35_PRIORITY	BYTE	2.0	分配的优先级：默认值 OB30 ~ OB38 分别是 7 ~ 15
OB35_OB_NUMBR	BYTE	3.0	OB 编号（30 ~ 38）
OB35_RESERVED_1	BYTE	4.0	保留
OB35_RESERVED_2	BYTE	5.0	保留
OB35_PHASE_OFFSET	WORD	6.0	相位偏移量（单位为 ms）
OB35_RESERVED_3	INT	8.0	保留
OB35_EXC_FREQ	INT	10.0	时间间隔（单位为 ms）
OB35_DATE_TIME	DATE_AND_TIME	12.0	调用 OB 时的 DATE_AND_TIME

【例】　通过 STEP 7 设置循环中断

1）建立完项目后出现的所有目录，如图 5-44 所示。接着在管理画面打开硬件组态画面，如图 5-45 所示。

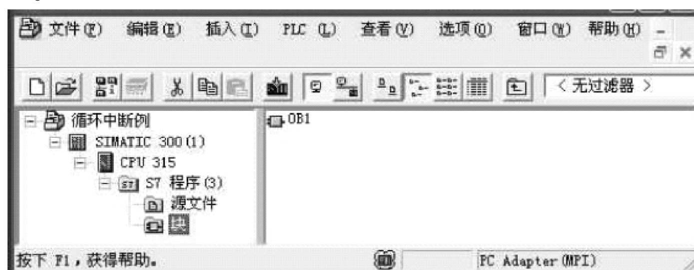


图 5-44　建立完成的所有目录



图 5-45　打开硬件组态

2）在硬件组态画面，双击机架上的 CPU，自动弹出 CPU 属性设置画面，如图 5-46 所示。

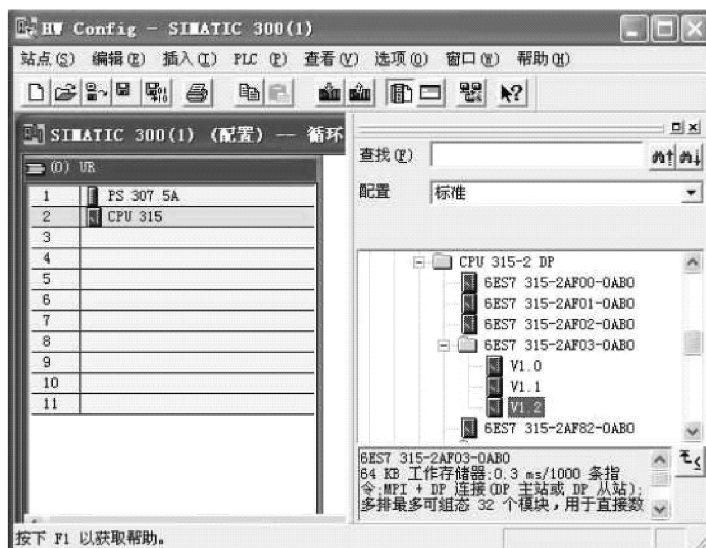


图 5-46　打开 CPU 属性设置界面

3）在 CPU 属性设置画面打开周期性中断设置画面，设置 OB35 循环中断执行时间为 3000ms，如图 5-47 所示，然后单击“确定”。

4）在管理画面的块目录里插入 OB35，如图 5-48 所示。打开 OB35 编写如图 5-49 所示的程序并保存。

5）最后在管理画面把整个项目的信息下载到 CPU（或 PLCSIM 仿真 CPU）中，运行 PLC，监控 OB35 程序状态，可以看到 MW10 的数值每秒钟加 2，如图 5-50 所示，表示 OB35 被激活了。

【例】　通过调用 SFC40 来设置循环中断，通过调用 SFC39 来取消循环中断。

1）在上例的基础上，在管理画面的块目录里打开 OB1，并在 OB1 编写如图 5-51 所示程序，然后保存。



图 5-47　CPU 属性中周期性中断的设置



图 5-48　在管理画面的块目录里插入 OB35

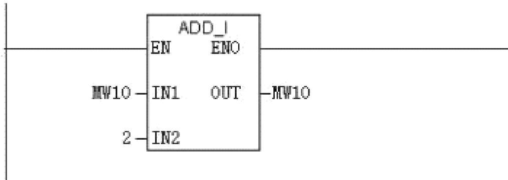


图 5-49　在 OB35 里编写的程序

2）最后在管理画面把整个项目的信息下载到 CPU（或 PLCSIM 仿真 CPU）中，运行 PLC，监控 OB35 程序状态，可以看到 MW10 的数值每秒钟加 2；当接通 M0.0 时，监控 MW10 的数值不再增加了，说明 OB35 循环中断被取消激活了；当接通 M0.2 时，可以看到 MW10 的数值又恢复了每秒钟加 2，说明 OB35 循环中断被重新激活了。



图 5-50　OB35 的程序状态

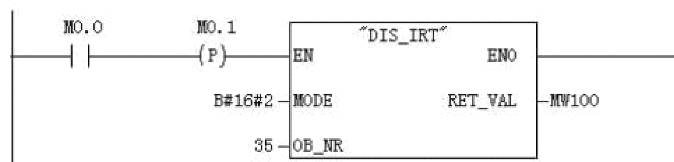
OB1 : "Main Program Sweep (Cycle)"循环中断

程序段 1: 取消激活OB35循环中断

通过SFC 39 "DIS_IRT" (禁用中断)，可以禁用新中断和异步错误的处理。
SFC 39 参数：

参数	声明	数据类型	存储器区	描述
MODE	INPUT	BYTE	I、Q、M、D、L、常数	指定禁用哪些中断和异步错误。
OB_NR	INPUT	INT	I、Q、M、D、L、常数	OB编号
RET_VAL	OUTPUT	INT	I、Q、M、D、L	如果在功能处于激活状态时出错，则返回值将包含错误代码。

MODE设置为B#16#2:禁用所有新发生的使用OB编号指定中断。继续在诊断缓冲区中输入条目。



程序段 2: 重新激活OB35循环中断

使用SFC 40 "EN_IRT" (启用中断)，可以启动通过40 "EN_IRT"激活的新中断和异步错误的处理。
SFC 40 参数：

参数	声明	数据类型	存储器区	描述
MODE	INPUT	BYTE	I、Q、M、D、L、常数	指定激活哪些中断和异步错误。
OB_NR	INPUT	INT	I、Q、M、D、L、常数	OB编号
RET_VAL	OUTPUT	INT	I、Q、M、D、L	如果在功能处于激活状态时出错，则返回值将包含错误代码。

MODE设置为B#16#2:激活所有新发生的使用OB编号指定中断。继续在诊断缓冲区中输入条目。

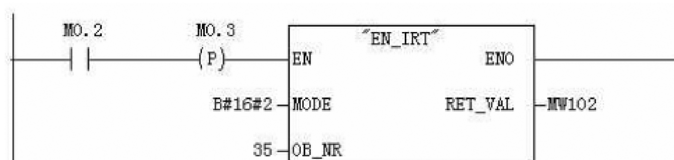


图 5-51　在 OB1 里编写的程序

5.2.6 同步循环中断组织块

同步循环中断 OB 是通过同步循环中断选择在具有 DP 循环的同步循环中启动的程序。OB61 充当同步循环中断 TSAL1 的接口 OB。可以将 OB61 的优先级设置为 0（取消选定 OB）和 2～26 之间的数。

在使用 L 或 T 命令直接访问以及使用 SFC14 “DPRD_DAT” 和 SFC15 “DPWR_DAT” 时，请避免访问已为其过程映像分区分配到 OB61～OB65 的连接的 I/O 区域。在 OB61～OB65 中系统定义了如表 5-9 所示的本地数据（表 5-9 中的符号以 OB61 为例。），其地址从 L0.0～L19.7，地址从 L20.0 以上的本地数据允许用户定义。

表 5-9　OB61 中系统定义的本地数据

符号名称	数据类型	地址	说　明
OB61_EV_CLASS	BYTE	0.0	事件等级和标识符 B#16#11：中断处于激活状态
OB61_STRT_INF	BYTE	1.0	B#16#64：OB 61 的启动请求 B#16#65：OB 62 的启动请求 B#16#66：OB 63 的启动请求 B#16#67：OB 64 的启动请求
OB61_PRIORITY	BYTE	2.0	分配的优先级；默认值为 25
OB61_OB_NUMBR	BYTE	3.0	OB 编号（61～64）
OB61_RESERVED_1	BYTE	4.0	保留
OB61_RESERVED_2	BYTE	5.0	保留
OB61_GC_VIOL	BOOL	6.0	GC 错误
OB61_FIRST	BOOL	6.1	启动或停止状态后的首次使用
OB61_MISSED_EXEC	BYTE	7.0	自上次执行 OB 61 以来启动 OB 61 失败的次数
OB61_DP_ID	BYTE	8.0	同步 DP 主站系统的 DP 主站系统 ID
OB61_RESERVED_3	BYTE	9.0	保留
OB61_RESERVED_4	WORD	10.0	保留
OB61_DATE_TIME	DATE_AND_TIME	12.0	调用 OB 时的 DATE_AND_TIME

技术同步中断 OB65 仅适用于 Technology CPU。通过技术同步中断可选择在更新技术块的同时启动程序。技术同步中断 OB 在更新技术块后启动。技术同步中断 OB 的优先级固定设置为 25，不能对其进行更改。

在 OB65 中系统定义了如表 5-10 所示的本地数据，其地址从 L0.0～L19.7，地址从 L20.0 以上的本地数据允许用户定义。

表 5-10　OB65 中系统定义的本地数据

符号名称	数据类型	地址	说　明
OB65_EV_CLASS	BYTE	0.0	事件等级和 ID：B#16#11：进入事件，事件等级 1
OB65_STRT_INF	BYTE	1.0	B#16#6A：OB 65 的启动请求

（续）

符号名称	数据类型	地址	说　　明
OB65_PRIORITY	BYTE	2.0	优先级:25(固定设置)
OB65_OB_NUMBR	BYTE	3.0	OB 编号(65)
OB65_RESERVED_1	BYTE	4.0	保留
OB65_RESERVED_2	BYTE	5.0	保留
OB65_RESERVED_3	BOOL	6.0	保留
OB65_FIRST	BOOL	6.1	启动后首次使用 OB 65
OB65_MISSED_EXEC	BYTE	7.0	自上次执行 OB 65 以来启动 OB 65 失败的次数
OB65_RESERVED_4	BYTE	8.0	保留
OB65_RESERVED_5	BYTE	9.0	保留
OB65_RESERVED_6	WORD	10.0	保留
OB65_DATE_TIME	DATE_AND_TIME	12.0	调用 OB 时的 DATE_AND_TIME

5.2.7　异步故障中断组织块

OB80 在执行程序过程中当出现超出周期时间、执行 OB 时出现确认错误、因提前时间而使 OB 的启动时间被跳过或在 CiR 后恢复 RUN 模式等任意一个错误，S7-300 CPU 的操作系统都会调用异步故障中断组织块 OB80。例如，如果在上一次调用之后发生了某一循环中断 OB 的启动事件，而同一 OB 此时仍在执行中，则操作系统将调用 OB80。可以使用 SFC39 ~ SFC42 禁用、延迟和重新启用时间错误 OB。如果因超出了扫描时间而导致在同一扫描周期内调用了两次 OB80，则 CPU 转为 STOP 模式。通过在程序中适当的位置调用 SFC43 “RE_TRIGR” 可防止这种情况。

如果在 OB80 中没有编写取消跳过的日期时间中断，只执行第一次跳过的日期时间中断，其他的被忽略了；如果需要执行新的日期时间中断，必须在 OB80 里编写判断是哪个日期时间中断，然后使用 SFC29 “CAN_TINT” 取消被跳过的日期时间中断。

在 OB80 中系统定义了如表 5-11 所示的本地数据，其地址从 L0.0 ~ L19.7，地址从 L20.0 以上的本地数据允许用户定义。

表 5-11　OB80 中系统定义的本地数据

符号名称	数据类型	地址	说　　明
OB80_EV_CLASS	BYTE	0.0	事件等级和标识符:B#16#35
OB80_FLT_ID	BYTE	1.0	错误代码:(可能值:B#16#01 ~ B#16#0B,共 11 种)
OB80_PRIORITY	BYTE	2.0	优先级:OB80 以优先级 26 在 RUN 模式下运行,如果发生 OB 请求缓冲区溢出,则以优先级 28 运行
OB80_OB_NUMBR	BYTE	3.0	OB 编号(80)
OB80_RESERVED_1	BYTE	4.0	保留

（续）

符号名称	数据类型	地址	说　　明
OB80_RESERVED_2	BYTE	5.0	保留
OB80_ERROR_INFO	WORD	6.0	错误信息;取决于错误代码
OB80_ERR_EV_CLASS	BYTE	8.0	导致错误的启动事件的事件等级
OB80_ERR_EV_NUM	BYTE	9.0	导致错误的启动事件的事件编号
OB80_OB_PRIORITY	BYTE	10.0	错误信息;取决于错误代码
OB80_OB_NUM	BYTE	11.0	错误信息;取决于错误代码
OB80_DATE_TIME	DATE_AND_TIME	12.0	调用 OB 时的 DATE_AND_TIME

第 6 章

顺序控制及S7 Graph编程

6.1 顺序控制设计法

6.1.1 顺序功能图

在顺序控制系统中，对于复杂顺序控制程序仅靠基本指令系统编程会感到很不方便，其梯形图复杂且不直观。西门子 S7-300/400 系列 PLC 为用户提供了顺序功能图语言（Sequential Function Chart，以下简称 SFC），用于编制复杂的顺序控制程序。利用这种编程方法能够较容易地编写出复杂的顺序控制程序，从而提高工作效率。

所谓顺序控制，就是按照生产工艺预先规定的顺序，在各个输入信号作用下，根据内部状态和时间的顺序，在生产过程中各个执行机构自动并有序地进行操作。使用顺序控制设计法时首先根据系统的工艺过程画出顺序功能图，然后根据顺序功能图画出梯形图。

顺序控制设计法是一种先进的设计方法，很容易被初学者接受；对于有经验的工程师，也会提高设计效率，节约大量的设计时间。其设计思想是将系统的一个工作周期划分为若干个顺序相连的阶段，这些阶段称为“步”（Step），并明确每一“步”所要执行的输出，“步”与“步”之间通过指定的条件进行转换。在程序中只需要通过正确连接进行“步”与“步”之间的转换，便可以完成系统的全部工作。

顺序控制程序与其他 PLC 程序在执行过程中的最大区别是：SFC 程序在执行程序过程中始终只有处于工作状态的“步”（称为“有效状态”或“活动步”）才能进行逻辑处理与状态输出，而其他状态的“步”（称为“无效状态”或“非活动步”）的全部逻辑指令与输出状态均无效。因此，使用顺序控制进行程序设计时，设计者只需要分别考虑每一“步”所需要确定的输出，以及“步”与“步”之间的转换条件，并通过简单的逻辑运算指令就可完成程序的设计。

顺序功能图主要由步、有向连线、转换、转换条件和动作（或命令）组成，如图 6-1 所示。

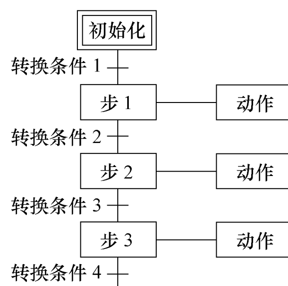


图 6-1 顺序功能图

1. 步

在顺序控制中步又称为状态，它是指控制对象的某一特定的工作情况。为了区分不同的状态，同时使得 PLC 能够控制这

些状态，需要对每一状态赋予一定的标记，这一标记称为状态元件。在西门子 S7 系列 PLC 中，状态元件通常用编程元件（例如存储器位 M）来表示。步主要分为初始步、活动步和非活动步。

初始状态一般是系统等待启动命令的相对静止的状态。系统在开始进行自动控制之前，首先应进入规定的初始状态。与系统的初始状态相对应的步称为初始步，初始步用双线框表示，每一个顺序控制功能图至少应该有一个初始步。

当系统处于某一步所在的阶段时，该步处于活动状态，称为活动步。步处于活动状态时，相应的动作被执行。处于不活动状态的步称为非活动步，其相应的非存储型动作被停止执行。

2. 有向连线

有向连线就是状态间的连接线，它决定了状态的转换方向与转换途径。在顺序控制功能图程序中的状态一般需要两条以上的有向连线进行连接，其中一条为输入线，表示转换到本状态的上一级“源状态”；另一条为输出线，表示本状态执行转换时的下一级“目标状态”。在顺序功能图程序设计中，对于自上而下的正常转换方向，其连接线一般不需标记箭头，但是对于自下而上的转换或是向其他方向的转换，必须以箭头标明转换方向。

3. 转换

步的活动状态的进展是由转换的实现来完成的，并与控制过程的发展相对应。转换用有向连线上与有向连线垂直的短线来表示，转换将相邻两步分隔开。

4. 转换条件

所谓转换条件是指用于改变 PLC 状态的控制信号。不同状态间的换转条件可以不同，也可以相同。当转换条件各不相同时，顺序功能图程序每次只能选择其中的一种工作状态（称为选择分支）。当若干个状态的转换条件完全相同时，顺序功能图程序一次可以选择多个状态同时工作（称为并行分支）。只有满足条件的状态，才能进行逻辑处理与输出，因此，转换条件是顺序功能图程序选择工作状态的开关。

在顺序功能图程序中，转换条件通过与有向连线垂直的短横线进行标记，并在短横线旁边标上相应的控制信号地址。

5. 动作

可以将一个控制系统划分为施控系统和被控系统。对于被控系统，动作是某一步所要完成的操作；对于施控系统，在某一步中要向被控系统发出某些“命令”，这些命令也可称为动作。在顺序功能图语言 S7-GRAPH 中，将动作的修饰词称为动作中的命令。

6.1.2 顺序功能图结构

在顺序功能图程序中，由于控制要求或设计思路的不同，步与步之间的连接形式也不同，从而形成了顺序功能图程序的 3 种不同基本结构形式：单序列、选择序列、并行序列。这 3 种序列结构如图 6-2 所示。

1. 单序列

单序列由一系列相继激活的步组成，每一步的后面仅有一个转换，每一个转换的后面只有一个步，如图 6-2a 所示。单序列结构的特点如下。

1）步与步之间采用自上而下的串联连接方式。

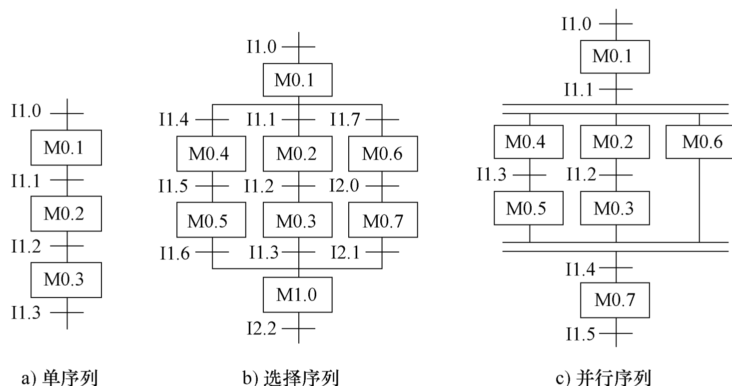


图 6-2　SFC 的 3 种序列结构图

2）状态的转换方向始终是自上而下且固定不变（起始状态与结束状态除外）。

3）除转换瞬间外，通常仅有一个步处于活动状态。基于此，在单序列中可以使用“重复线圈”（如输出线圈、内部辅助继电器等）。

4）在状态转换的瞬间，存在一个 PLC 循环周期时间的相邻两状态同时工作的情况，因此对于需要进行互锁的动作，应在程序中加入互锁触点。

5）在单序列结构的顺序功能图程序中，原则上定时器也可以重复使用，但不能在相邻两状态里使用同一定时器。

6）在单序列结构的顺序功能图程序中，只能有一个初始状态。

2. 选择序列

选择序列的开始称为分支，如图 6-2b 所示，转换符号只能标在水平连线之下。

在图 6-2b 中，如果步 M0.1 为活动步且转换条件 I1.1 有效时，则发生由步 M0.1→步 M0.2 的进展；如果步 M0.1 为活动步且转换条件 I1.4 有效时，则发生由步 M0.1→步 M0.4 的进展；如果步 M0.1 为活动步且转换条件 I1.7 有效时，则发生由步 M0.1→步 M0.6 的进展。

在步 M0.1 之后选择序列的分支处，每次只允许选择一个序列。选择序列的结束称为合并，几个选择序列合并到一个公共序列时，用与需要重新组合的序列以相同数量的转换符号和水平连线来表示，转换符号只允许标在连线之上。

允许选择序列的某一条分支上没有步，但是必须有一个转换，这种结构的选择序列称为跳步序列。跳步序列是一种特殊的选择序列。

3. 并行序列

并行序列的开始称为分支，如图 6-2c 所示，当转换的实现导致几个序列同时激活时，这些序列称为并行序列。

在图 6-2c 中，当步 M0.1 为活动步时，若转换条件 I1.1 有效，则步 M0.2、步 M0.4 和步 M0.6 均同时变为活动步，同时步 M0.1 变为不活动步。为了强调转换的同步实现，水平连线用双线表示。步 M0.2、步 M0.4 和步 M0.6 被同时激活后，每个序列中活动步的进展将是独立的。在表示同步的水平双线上，只允许有一个转换符号。并行序列用来表示系统的几

个同时工作的独立部分的工作情况。

用 S7 Graph 编写的一个顺序控制项目至少需要 3 个块（见图 6-3）。

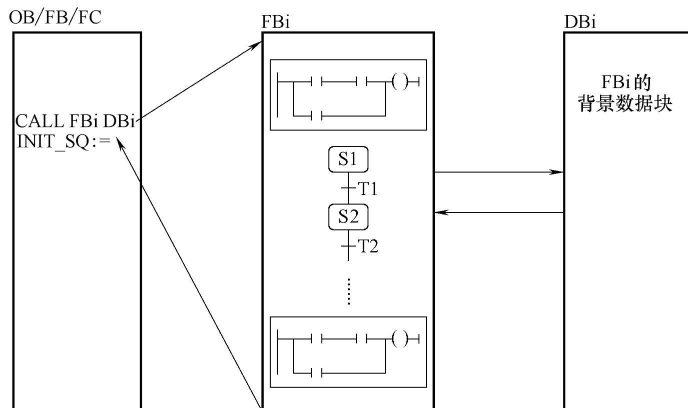


图 6-3 顺序控制系统中的块

1）一个调用 S7 Graph FB 的块，它可以是组织块（OB）、功能块（FB）或功能（FC）；

2）一个用来描述顺序控制系统各子任务（步）和相互关系（转换）的 S7 Graph FB，它由一个或多个顺序控制器（Sequencer）组成；

3）一个指定给 S7 Graph FB 的背景数据块（DB），它包含了顺序控制系统的参数。

用 S7 Graph 编写的顺序控制功能图程序是由 FB 和 DB 组成的，并被主程序 OB1 调用。FB 用于执行顺序功能图程序的控制功能，包含许多系统定义的参数，并通过参数设置来对整个顺序系统进行控制，从而实现系统的初始化和工作方式的转换等功能；DB 是顺序控制功能块 FB 的背景数据块，并包含顺序功能图程序结构和相关数据。

调用 S7 Graph FB 时，顺序控制器从第一步或从初始步开始启动。一个 S7 Graph FB 可以包含多个步，最多可以包含 250 步。在线性顺序控制器中，在初始步之后交错排列的转换和步，一个 S7 Graph FB 中最多可以包含 250 个转换。

除线性时序（每步之后仅有单个步）外，顺序控制器还可以有多个分支。一个顺序控制器最多包含 256 个分支、249 条并行序列的分支和 125 条选择序列的分支。对于每个 PLC 而言，分支数并不是完全相同的，它与 CPU 的型号有关。通常，只能用 20～40 条分支，如果使用的分支数过多，则会使程序的执行时间特别长。可以在路径结束时，在转换之后添加一个跳转（Jump）或一个支路的结束点（Stop）。结束点将正在执行的路径变为不活动的路径。

6.2 S7 Graph 顺序功能图编程

6.2.1 认识 S7 Graph 编程环境

在本节中，我们以红绿灯动作控制为例，介绍 S7 Graph 的功能，如图 6-4 所示为红绿灯动作控制示意图。

1. 创建项目及硬件组态

1）首先创建一个目录并完善目录的硬件组态，然后在块的目录里插入编写语言为 GRAPH 的功能块。

操作步骤为：单击“插入”→“S7 块”→“功能块”，如图 6-5 所示。

2）在插入功能块的过程中，自动弹出咨询设置功能块属性的对话框，如图 6-6 所示。

在创建语言栏里选择“GRAPH”，如果有必要同时可以填写其他的一些附加信息，然后单击“确定”。

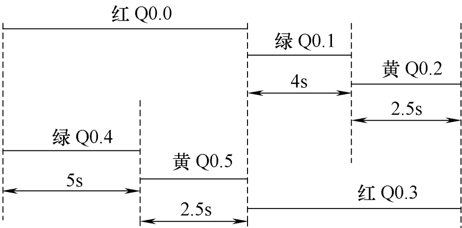


图 6-4　红绿灯动作时序图

Q0.0、Q0.1 和 Q0.2 代表东西方向的交通灯；
Q0.3、Q0.4 和 Q0.5 代表南北方向的交通灯。



图 6-5　S7 项目管理界面



图 6-6　声明 FB1 的编程语言使用 GRAPH

3）在块目录里可以看到刚插入的功能块（本例是 FB1），双击 FB1 的图标打开 FB1 的编程界面，自动创建了第 1“步”（S1）和第 1 个“转移”（Trans1），如图 6-7 所示。

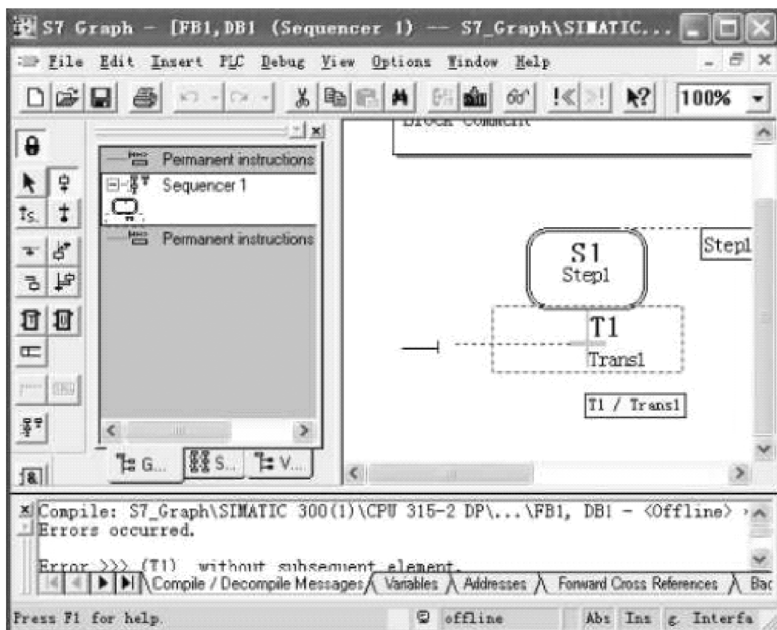


图 6-7　插入“步”



提示：

浏览窗口里可以查看 Graphics（图形）、Sequencer（顺序器）和 Variables（变量）3 种浏览界面，如图 6-8 所示。

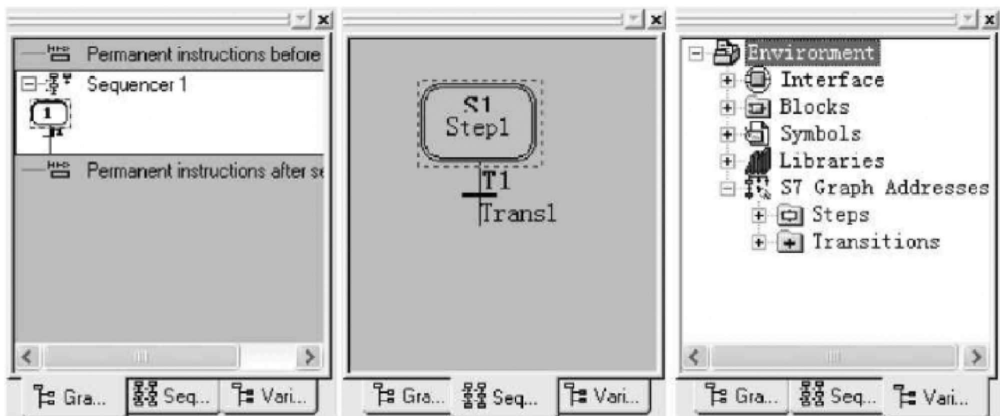


图 6-8　浏览窗口的 3 种界面

Graphics（图形）浏览窗上面和底下是永久程序界面，中间是分层的 Graph 程序界面。

Sequencer（顺序器）浏览窗可以浏览程序的总体结构，同时可以浏览程序界面的局部内容。

Variables（变量）浏览窗可以浏览到编程时可能用到的元素，在这里除系统变量外，可

以定义、编辑及修改变量。

一个 S7 Graph FB 最多可以编写 250 个“步”和 250 个“转换”。当 S7 Graph FB 激活时总是从最顶的“步”或从初始“步”开始执行。

一个 S7 Graph FB 可以编写多个 Sequencer（顺序器）。每个 Sequencer（顺序器）最多可以编写 256 个分支、249 个并行分支及 125 个选择分支，具体容量与 CPU 型号有关。

一个顺序控制项目至少由 S7 Graph FB（可以是 OB、FC、SFC、FB、SFB）、Sequencer（可以有多个顺序器）和 DB（数据块，包括 SDB）组成。

2. 创建时序动作“步”

1）再回来分析图 6-4 所示的红绿灯时序动作，可以使用 4 个“步”来完成控制。在 FB1 里增加“步”，方法是使用鼠标单击“Trans1”附近（虚线框大小的范围），然后单击“ ”符号，即可以在指定地方插入一个“步”和一个“转移”，如图 6-7 所示。

2）S7 Graph 在进行插入操作中，允许两种模式：Direct（直接）模式和 Drag-and-Drop（拖放）模式。

选择 Direct 模式：单击“Insert”→“Direct”表示选择直接模式。也可以单击 Sequencer”工具条的“ ”图标让其凸上来，表示选择直接模式，如图 6-9 所示。

使用 Direct（直接）模式：首先在界面选择准备插入的位置，然后单击图 6-9 所示希望插入的目标图标，即可以在指定位置插入期待的目标。如果是插入多个相同的目标，可以连续单击目标的图标，每单击一次就插入一个。

选择 Drag-and-Drop（拖放）模式：单击“Insert”→“Drag-and-Drop”表示选择拖放模式。也可以单击“Sequencer”工具条的“ ”图标让其凹下去，表示选择拖放模式，如图 6-9 所示。

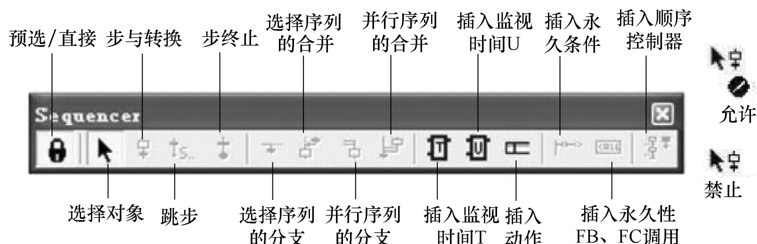


图 6-9　“Sequencer”工具条

插入“步”的动作框：首先选择插入模式“直接”或“拖放”，然后单击“Insert”→“Action”或“ ”图标，即可以插入“步”的动作框，如图 6-10 所示。

3）在动作框里编辑控制动作，每一个动作框包含指令和地址。比如在动作框左边写上指令“N”，在右边写上地址“Q0.0”，表示当该“步”为活动步时 Q0.0 输出“1”，当该“步”为不活动步时 Q0.0 输出“0”，如图 6-11 所示。

4）按照同样方法把每一“步”的动作框编写完整，如图 6-12 所示。

动作框里常用的指令见表 6-1。

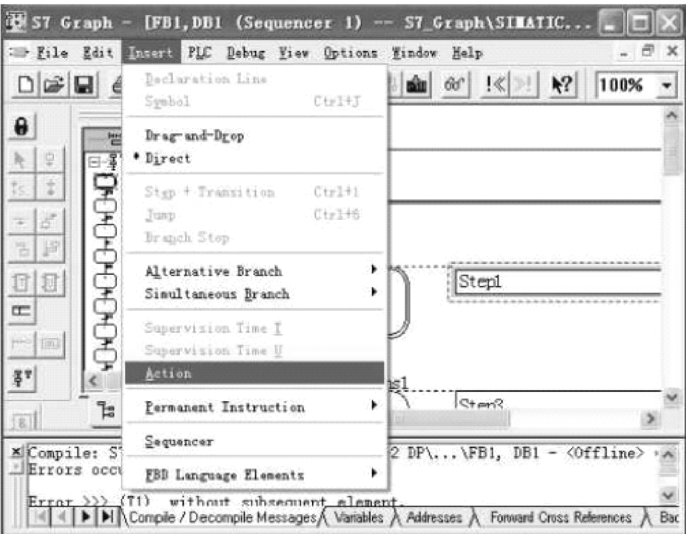


图 6-10 增加“步”

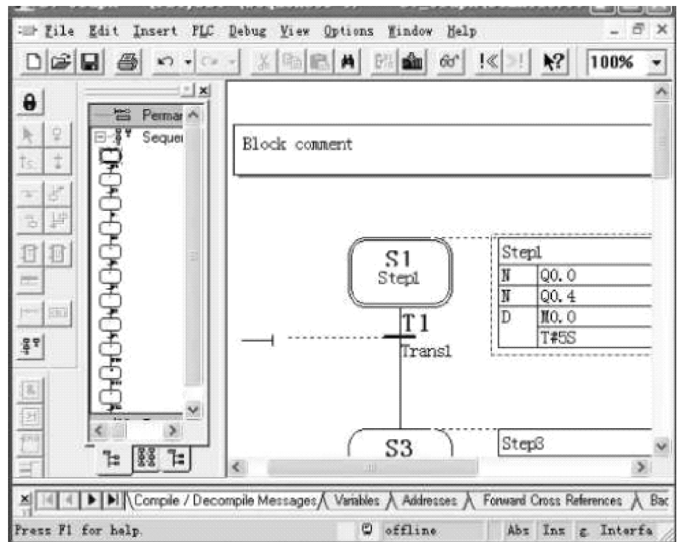


图 6-11 编辑“步”

表 6-1 动作框常用指令

指令(符号)	指令基本动作描述
N	当该“步”为活动步时地址输出“1”;当该“步”为不活动步时地址输出“0”
S	当该“步”为活动步时地址输出“1”并保持(即置位)
R	当该“步”为活动步时地址输出“0”并保持(即复位)
D	当该“步”为活动步时,开始计时(时间由该框 T#xx 指定),当时间到地址输出“1”; 当该“步”为不活动步时地址输出“0”
L	当该“步”为活动步时,地址输出“1”并开始计时(时间由该框 T#xx 指定),当时间到地址输出“0”; 当该“步”为不活动步时地址输出“0”
CALL	当该“步”为活动步时,调用指定的程序块

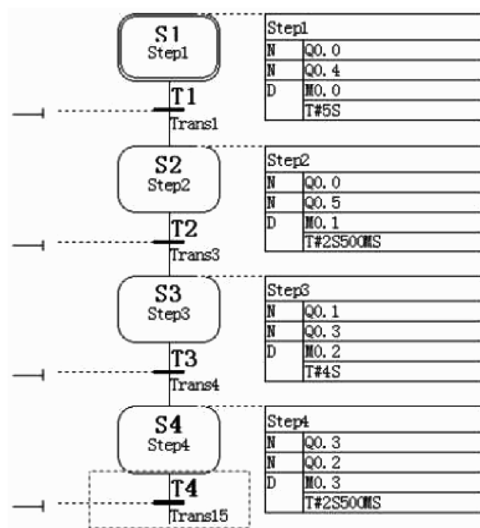


图6-12 “步”动作

5）编写转移条件的程序，转移条件程序的编写语言有LAD（梯形图）和FBD（功能图块）两种，单击“View”→“LAD”或“FBD”可以在这两种语言中互相切换，如图6-13所示，本例选择LAD梯形图。

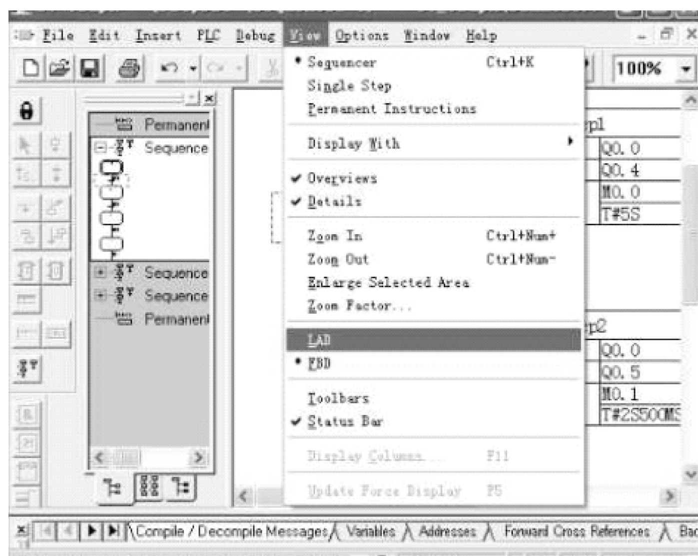


图6-13 选择编辑转移条件程序的语言

转移条件程序的指令一般可以是常开触点、常闭触点、比较指令、监视时间T或监视时间U，在顺序器工具条中分别用“┤├”、“┤/├”、“ ”、“ ”和“ ”表示（选择梯形图编程语言时）。

插入转移“指令”：首先选择插入模式“直接”或“拖放”，然后单击“┤├”、“┤/├”、“ ”、“ ”或“ ”图标，即可以在指定地方插入转移指令。然后在每个指令的地方

写上地址即可。比如选择“直接”模式，选中“步 1”（S1）的转移指令地方，再单击“┤├”就可以把常开触点指令放到“步 1”（S1）的转移指令里，如图 6-14 所示，然后写上指令的地址“M0.0”，如图 6-15 所示。

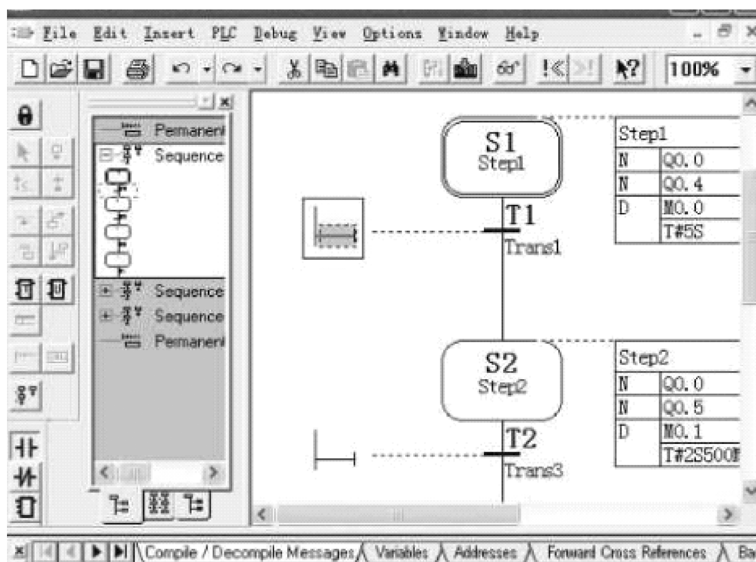


图 6-14　写上转移条件的程序指令

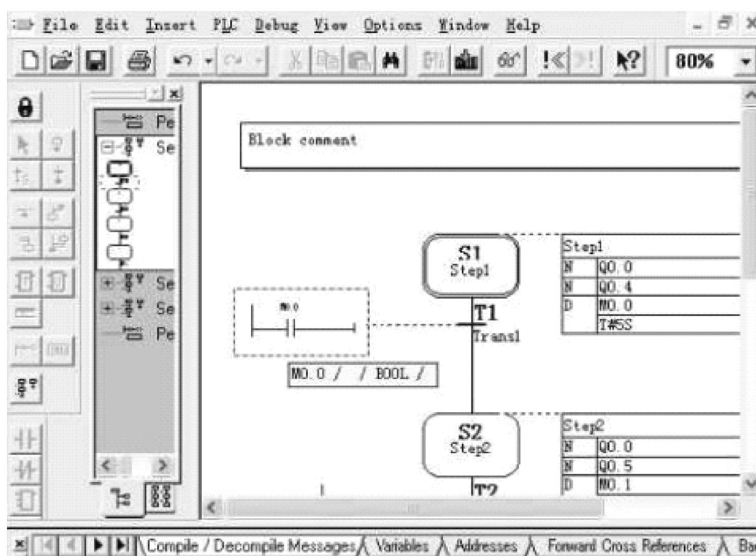


图 6-15　写上指令的软元件地址

6）使用同样方法编写其他步的转移指令，如图 6-16 所示。

顺序器中“步”的最后一般是跳转或结束指令，在顺序器工具条中分别用“↕S”和“↧”表示。

插入跳转或结束“指令”：首先选择插入模式“直接”或“拖放”，然后单击“↕S”或“↧”图标，即可以在指定地方插入跳转或结束指令。如果是跳转指令还需要写上跳转到那

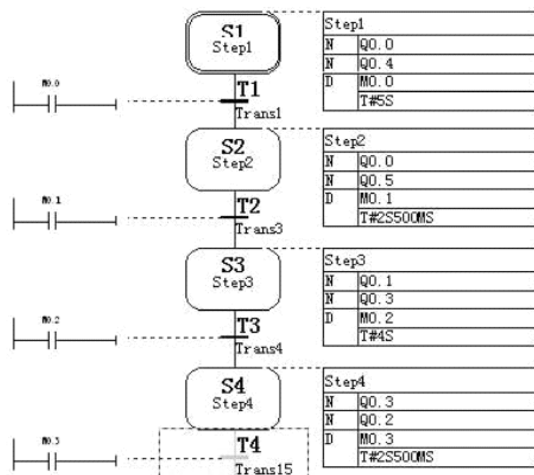


图 6-16　完成各“步”转移条件的程序

一“步”的地址代码。如果选择“直接”模式，选中“步 4”（S4）的跳转或结束指令地方，再单击“”就可以把跳转指令放到“步 4”（S4）的跳转或结束指令里，如图 6-17 所示，然后写上指令的地址“S1”，如图 6-18 所示，意思是要求完成“步 4”（S4）后跳转到“步 1”（S1），完成一个动作周期后开始下一个动作周期。

7）在 FB1 程序编辑界面，单击“”进行编译和保存，如图 6-19 所示。自动调用标准及系统功能如图 6-20 所示。在管理界面块目录里双击打开 OB1，在 OB1 里无条件调用 FB1，声明 DB1 为 FB1 的背景数据，在 FB1 里写上实形变量，如图 6-21 所示，单击“”进行编译及保存 OB1。

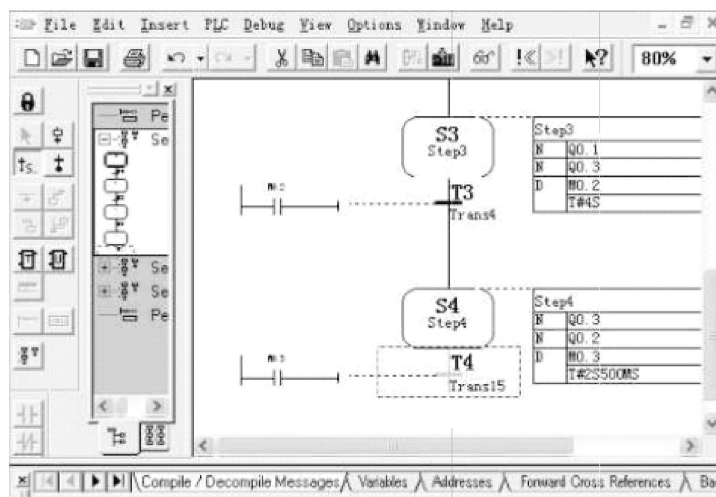


图 6-17　插入跳转或结束

3. 调试

1）打开 PLCSIM（或实际的 PLC），在管理界面里把站的所有信息下载到 CPU 中。在 PLCSIM 界面里打开 QB0、MB0 及 MB1 的垂直变量窗口，并单击“Tools”→“Options”→“At-

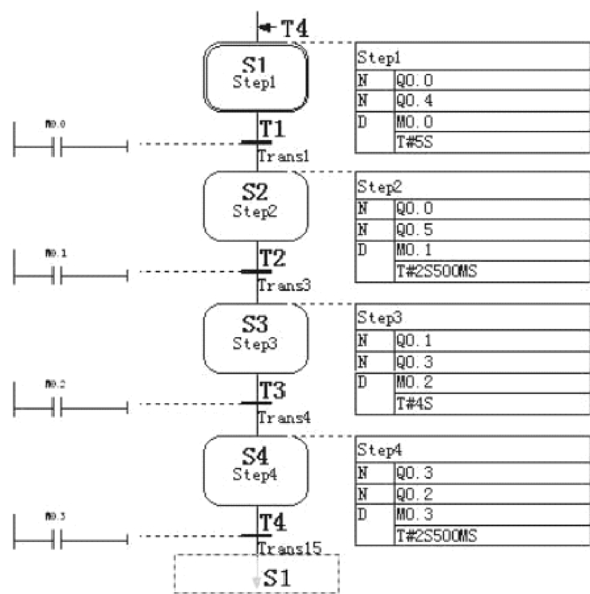


图 6-18 完整的红绿灯各“步”程序

tach Symbols”，按照保存的路径打开该项目的符号表并双击符号表，符号信息即可以显示在 PLCSIM 界面了，如图 6-22 所示。

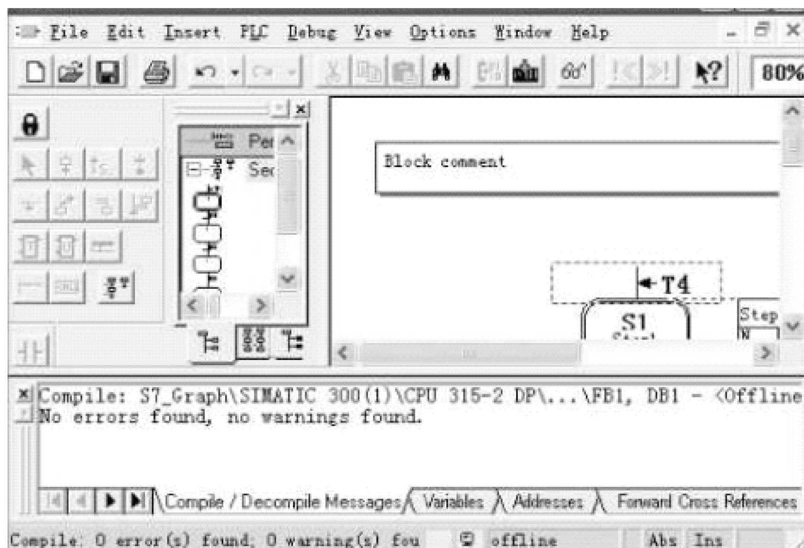


图 6-19 保存编译 FB1

2）现在开始调试程序。接通 M0.5 关闭 FB1 的顺序器，使所有的“步”都变为不活动的“步”，再接通 M1.0 准备自动连续执行（M1.4 自动灯亮起来了），断开 M0.5 接通 M0.6 激活 FB1 的顺序器，可以看到交通灯按照图 6-4 所示的时序图亮起来了。

3）现在进行单步调试。接通 M0.5 关闭 FB1 的顺序器，使所有的“步”都变为不活动，再接通 M1.1 准备自动单步执行（M1.5 单步灯亮起来了），断开 M0.5 接通 M0.6 激活 FB1 的顺序器，可以看到 Q0.0 和 Q0.4 亮起来了，5s 后 M0.0 也亮起来了；这时接通 M1.2



图 6-20　自动调用标准及系统功能

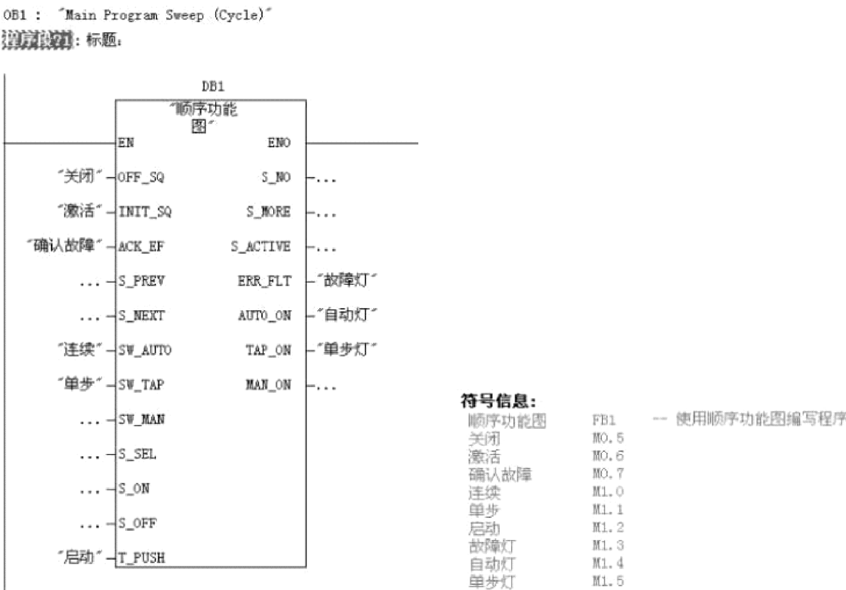


图 6-21　在 OB1 里调用 FB1

马上转移到下一步执行，可以看到 Q0.0 和 Q0.5 亮起来了，2.5s 后 M0.1 也亮起来了；这时接通 M1.2 马上转移到下一步执行，可以看到 Q0.1 和 Q0.3 亮起来了，4s 后 M0.2 也亮起来了；这时接通 M1.2 马上转移到下一步执行，可以看到 Q0.2 和 Q0.3 亮起来了，2.5s 后 M0.1 也亮起来了；再接通 M1.2 马上转移到下一个动作周期执行。

6.2.2　了解 S7 Graph 调试方法

在本节中我们仍以 6.2.1 中的例子继续介绍。

1）在管理界面把项目信息下载到 CPU 中，然后打开块目录的 FB1。在 FB1 界面里单击菜单栏的"Debug"（调试）→"Control Sequencer"，自动弹出图 6-23 所示的控制顺序器界面。

2）在控制顺序器界面，有 4 种调试模式：自动、手动、单步和自动/切换到下一步。当 CPU 在 RUN 状态时只能在自动调试模式；当在 RUN-P 状态时，可以在自动、手动、单

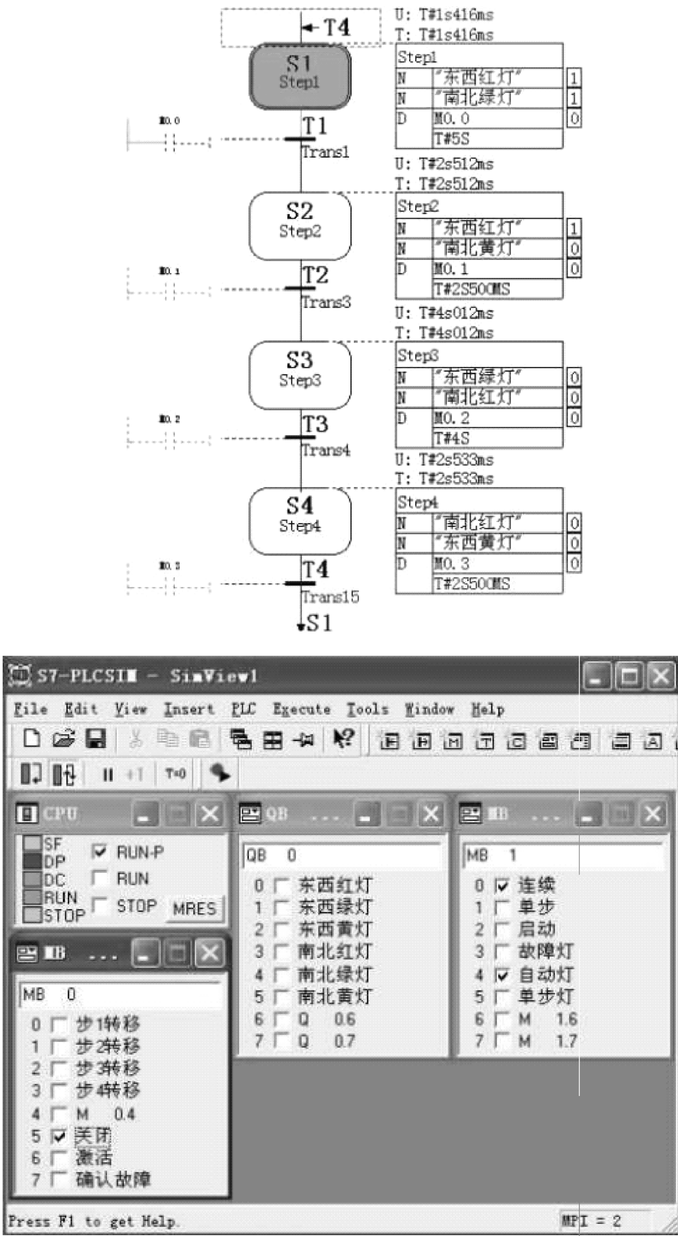


图 6-22　调试程序

步 3 种模式之间互相切换。

（1） 自动调试模式

自动调试模式的控制顺序器界面如图 6-24 所示。

选上控制顺序器界面“Automatic”前面的复选框，表示选择自动调试模式，同时确认被挂起的所有错误。这时可以看到交通灯按照图 6-4 所示的时序图亮起来了。

当单击“Initialize”按钮时，除了现在是步 1 被激活（初始步）外，其他任意已激活的步将会被取消激活，并重新启动顺序器激活初始步。

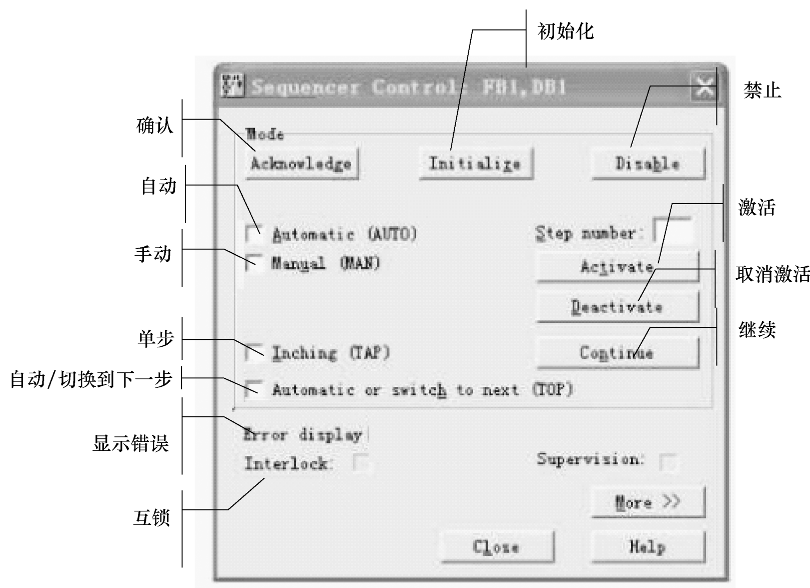


图 6-23　控制顺序器界面

当单击“Disable”按钮时，所有步将取消激活，要重新启动顺序器，可以单击“Initialize”按钮，重新启动顺序器激活初始步。

（2）手动调试模式

手动调试模式的控制顺序器界面如图 6-25 所示。



图 6-24　自动调试模式的控制顺序器界面



图 6-25　手动调试模式的控制顺序器界面

选中控制顺序器界面“Manual”前面的复选框，表示选择手动调试模式。在开始手动调试前，一般先单击“Disable”按钮禁止顺序器，所有步被取消激活。

在“Step number”栏里写上准备激活的“步”的编号，然后单击“Activate”按钮激活

指定的步。在单序列的顺序器中，只允许一个步激活，所以在激活其他步时，必须将已经激活的步取消激活，单击“Deactivate”按钮即可以取消激活当前被激活的步。

比如在“Step number”栏里写上“2”，然后单击“Activate”按钮，步 2 变为激活；单击 Deactivate”按钮步 2 被取消激活，在“Step number”栏里写上“4”，然后单击“Activate”按钮，步 4 变为激活。当单击“Initialize”按钮时，除了现在是步 1 被激活（初始步）外，其他任意一步被激活状态将会被取消激活，并重新启动顺序器激活初始步。

（3）单步调试模式

单步调试模式的控制顺序器界面如图 6-26 所示。

选中控制顺序器界面“Inching”前面的复选框，表示选择单步调试模式。在开始手动调试前，一般先单击“Disable”按钮禁止顺序器，所有步被取消激活。

然后单击“Initialize”按钮，重新启动顺序器激活初始步，可以看到 Q0.0 和 Q0.4 亮起来了，5s 后 M0.0 也亮起来了；这时单击“Continue”按钮马上转移到下一步执行，可以看到 Q0.0 和 Q0.5 亮起来了，2.5s 后 M0.1 也亮起来了；这时单击“Continue”按钮马上转移到下一步执行，可以看到 Q0.1 和 Q0.3 亮起来了，4s 后 M0.2 也亮起来了；这时单击“Continue”按钮马上转移到下一步执行，可以看到 Q0.2 和 Q0.3 亮起来了，2.5s 后 M0.1 也亮起来了；再单击“Continue”按钮马上转移到下一个动作周期执行。



图 6-26　单步调试模式的控制顺序器界面

（4）自动/切换到下一步调试模式

在自动/切换到下一步调试模式时，如果转移条件满足，将会自动转移到下一步；如果转移条件不满足，单击“Continue”按钮也可以转移到下一步。

（5）相关显示和附属参数设置

当设置了互锁和监控时，如果出现互锁错误或监控错误，显示窗指示灯会变为红色；否则是绿色。

单击“More”按钮可以打开附属参数设置窗口，在里面可以设置其他的一些调试参数。

6.2.3　分支程序的应用

在本节开头，用一个红绿灯程序引入，介绍了 S7 Graph 的基本知识，接下来，利用红绿灯程序讲解时序动作控制功能，下面介绍并行分支与合并程序的编程方法。

1）新建名为“交通灯”的项目，并完善项目的硬件组态，如图 6-27 所示，同时插入一个编程语言为“GRAPH”的功能块（本例是 FB1）和 FB1 的背景数据块 DB1。

2）打开 FB1 并在里面插入单序列的 5 个步，如图 6-28 所示。

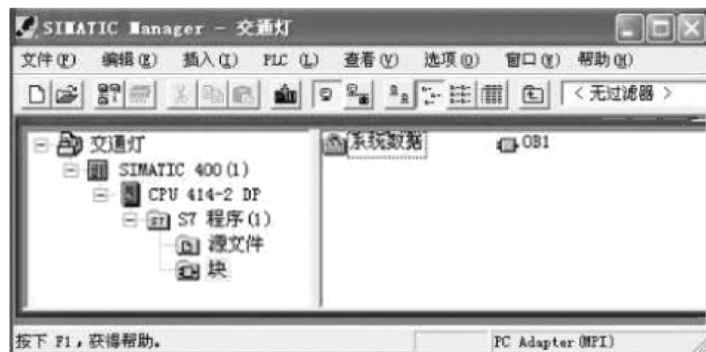


图 6-27　建立新项目

3）然后选中（单击鼠标左键）需要插入并行分支的地方（如图中 T1 附近的地方），然后单击“ ”即可以插入一个并行分支的“步”，如图 6-29 中的“S8”。

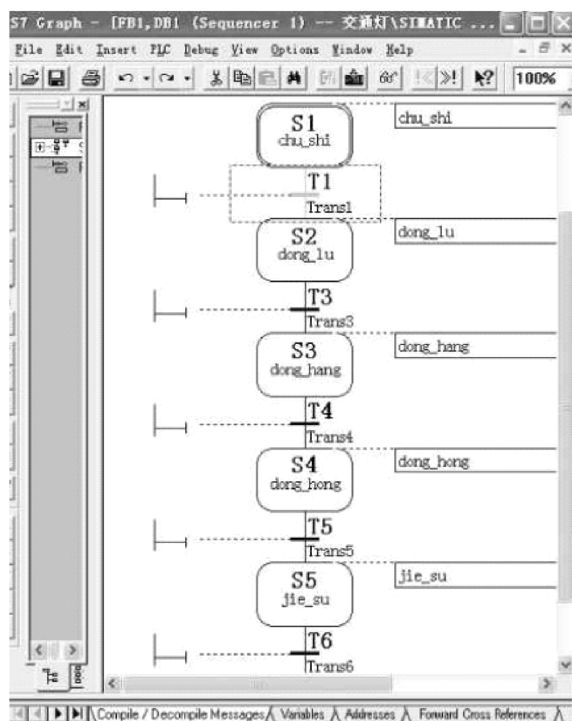


图 6-28　插入并行分支

4）插入并行合并的“步”。选中步“S8”，单击“ ”，这时鼠标出现一个附加的光标，把附加的光标拖动到准备合并的地方（本例是 T4 附近），如图 6-29 所示，然后单击鼠标左键即可。

5）利用同样的方法插入另外一个并行分支和合并的“步”，如图 6-30 所示的“S9”和“S10”。

6）在各个“步”按照系统要求编写动作程序和转移条件程序，如图 6-31 所示。

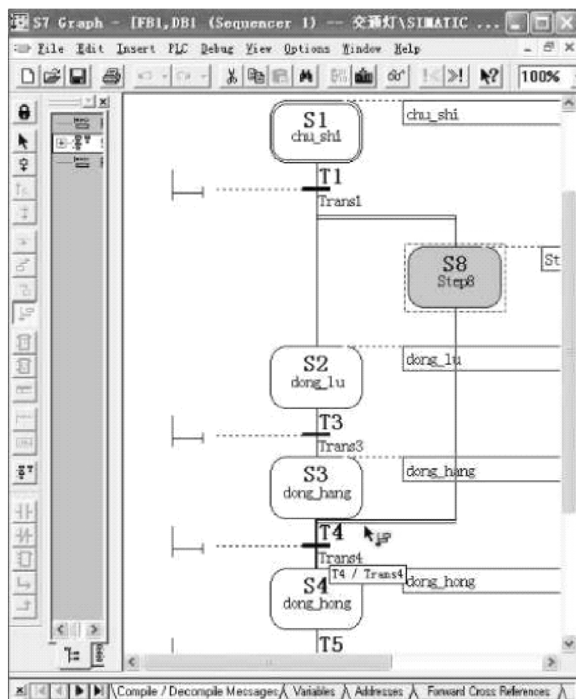


图 6-29 插入并行合并

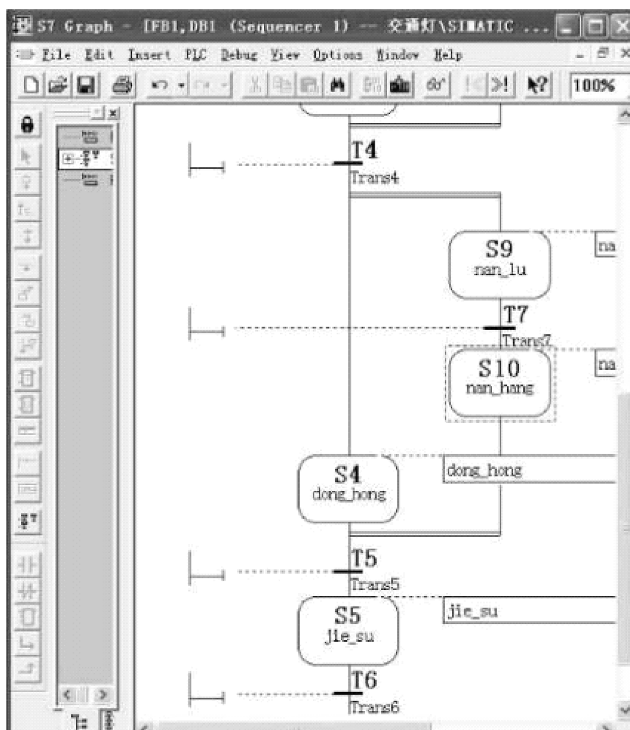


图 6-30 编写另外一个并行的分支和合并的“步”

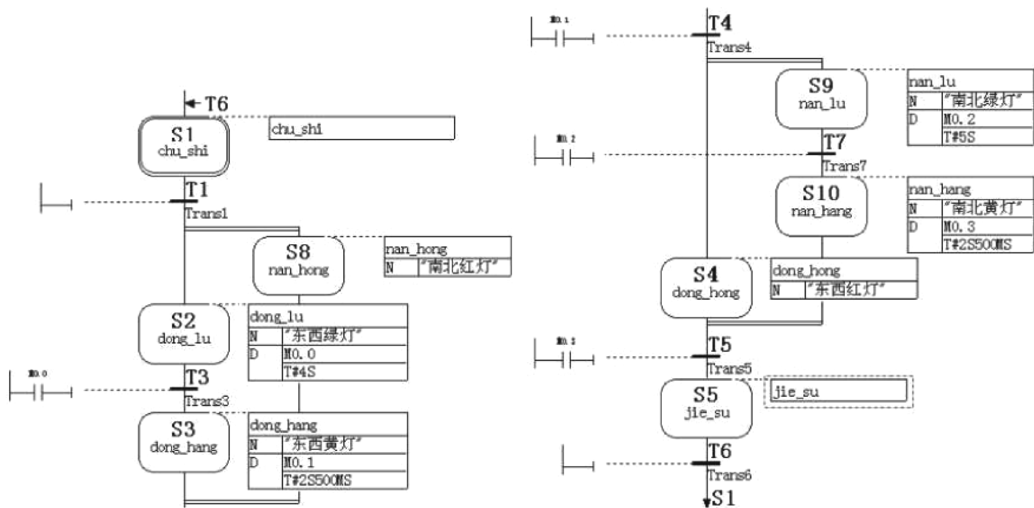


图 6-31　编写动作和转移程序

7）图 6-31 的“S1”和“S5”（空步）中没有动作，不需要编写动作程序，同时转移条件是无条件转移，也不需要编写。由于在“S1”和“S5”中没有动作和转移程序，在保存编译时会出现警告提示，这没有问题，这些警告只是起提醒作用。

保存编译结束时在信息栏出现的编译结果字样如下：

Compile：交通灯\SIMATIC 400（1）\CPU 414-2 DP\...\FB1，DB1- <Offline>

Warnings occurred

Warning >>>（T1）without contents.

Warning >>>（T1，Condition）Condition is always true.

Warning >>>（T6）without contents.

Warning >>>（T6，Condition）Condition is always true.

Warning >>>（S1）without contents.

Warning >>>（S5）without contents.

0 error（s）found；6 warning（s）found

8）在 OB1 里编写调用 FB1 的程序，如图 6-32 所示。

9）然后在项目管理界面把整个项目信息下载到 CPU 中（本例是 PLCSIM），然后运行 CPU，打开 FB1 监控程序运行，如图 6-33 所示。

6.2.4　实例演示多个顺序器程序

本例子完成图 6-4 所示红绿灯的时序动作控制功能，下面介绍多个顺序器程序的编程方法。同时为了突出黄灯的提醒效果，增加闪烁功能。

1）新建名为“顺序器”的项目，并完善项目的硬件组态，同时插入一个编程语言为“GRAPH”的功能块（本例是 FB10）和 FB10 的背景数据块 DB10。

2）在块目录里双击 FB10 打开 FB10 的编程界面，首先在 FB10 里增加 4 个单序列的“步”（比如分别是 S1、S2、S3 和 S4），然后在 FB1 程序界面的空白处单击鼠标右键，在

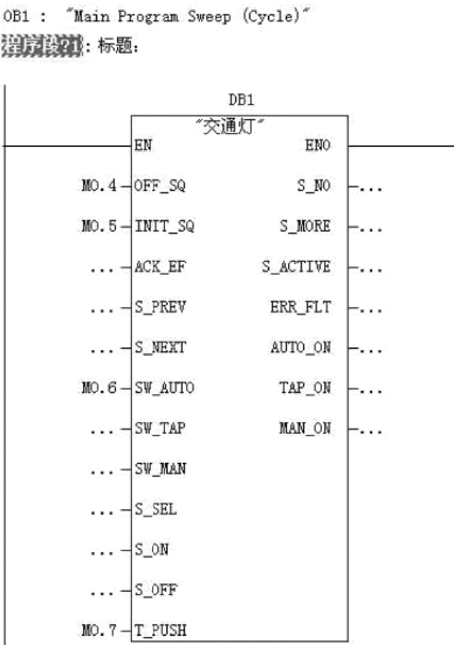


图 6-32　在 OB1 里编写的程序

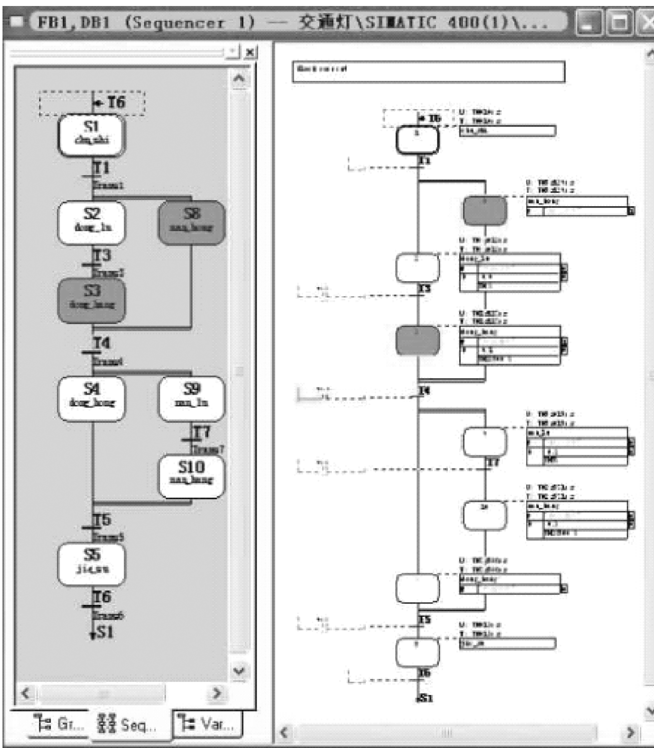


图 6-33　FB1 程序监控界面

弹出的对话框中单击"Insert New Element"→"Sequencer"，增加一个顺序器"Sequencer 2"，如图 6-34 所示；利用同样方法增加一个顺序器"Sequencer 3"。

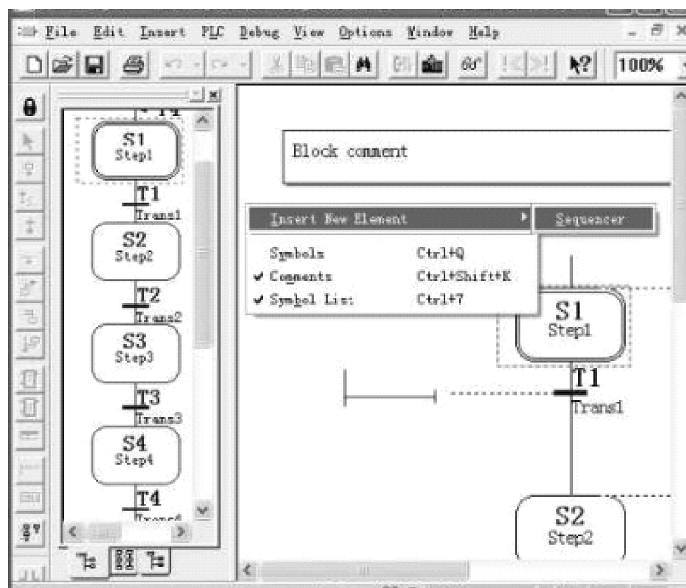


图 6-34　增加顺序器

3）单击浏览视图的 Sequencer 2，打开步 5（S5）的编程界面，编写图 6-35 所示的程序。

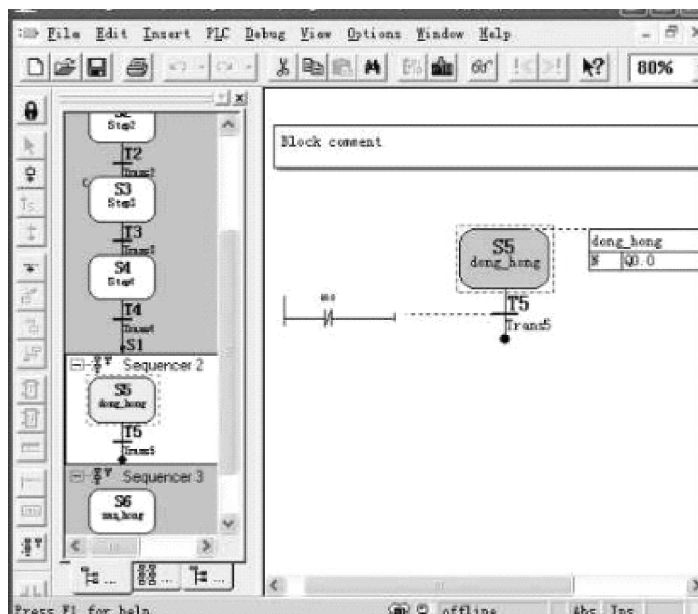


图 6-35　在 Sequencer 2 里编写程序

4）单击浏览视图的 Sequencer 3，打开步 6（S6）的编程界面，编写图 6-36 所示的程序。

5）在块目录里插入一个功能 FC1，并在里面编写黄灯闪烁的控制逻辑程序，如图 6-37 所示。

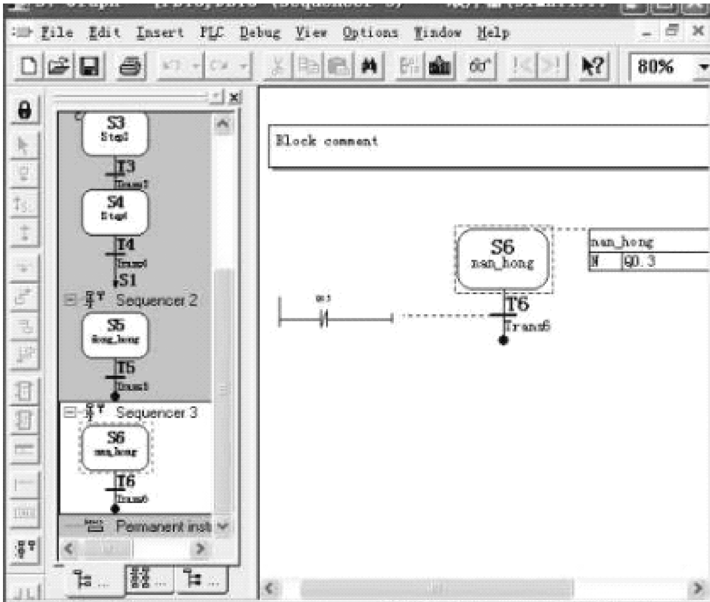


图 6-36 在 Sequencer 3 里编写程序

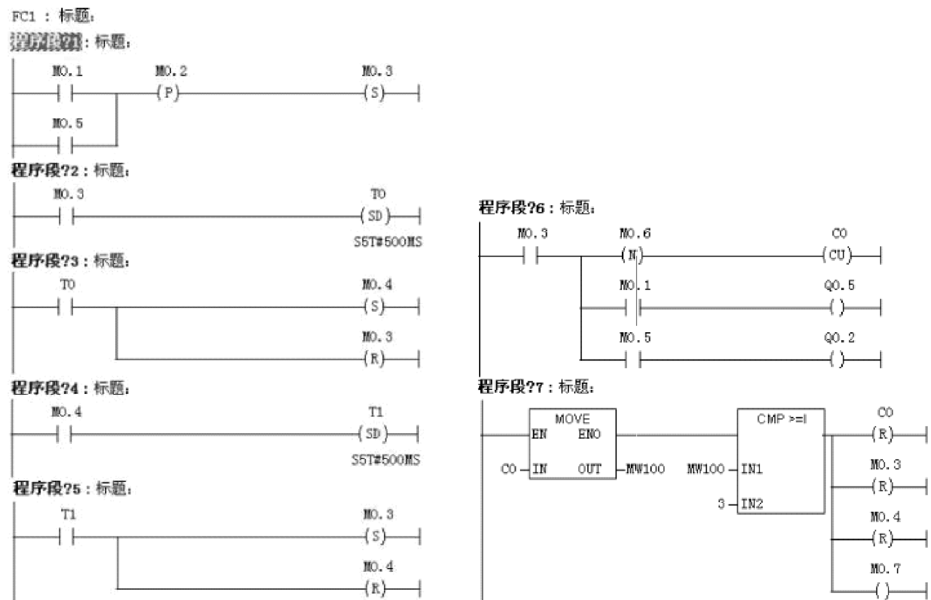


图 6-37 在 FC1 里编写的程序

与一个“步”直接有关系的程序有：动作、转换条件、监控和互锁程序。下面介绍这些程序的编程方法。

6）编写步 1（S1）的程序。在 FB1 的 Sequencer 1 界面，选中步 1（S1），单击“View”→“Single Step”，打开“步”的单步编程界面，如图 6-38 所示，并在里面编写图 6-39 所示的程序。

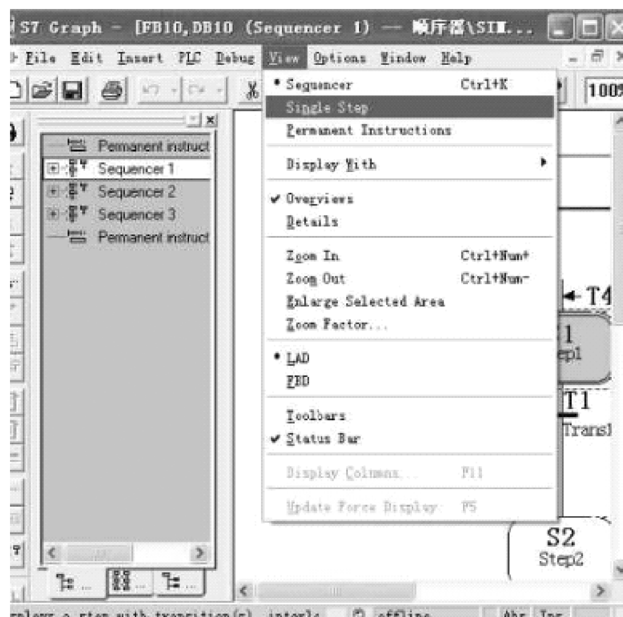


图 6-38　打开“步”的单步编程界面

“Interlock”（互锁）的功能是当互锁条件不满足时，尽管 S1 是激活的，该步与互锁有关的动作也是没有输出的，同时该步在监控时是红色的；只有当互锁条件满足，该步在激活时与互锁有关的功能才会执行。

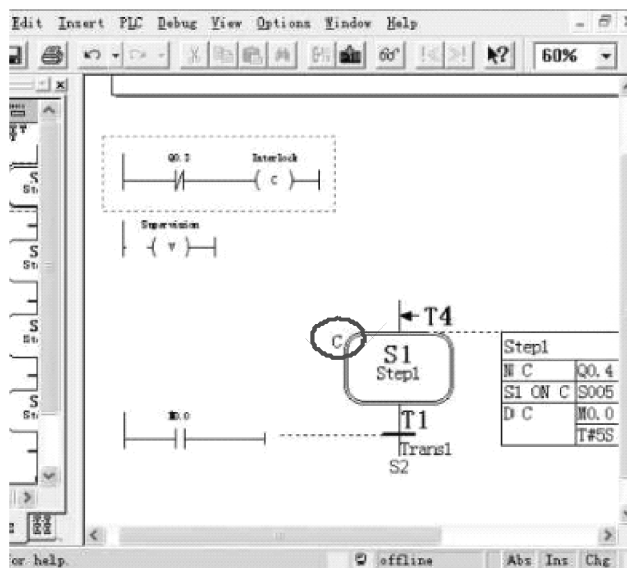


图 6-39　编入步 1 的程序

7）S1 步的图标左上角上有一个“C”的标志，表示该步设置了互锁条件，在动作程序中，与互锁相关的动作指令都带“C”字母。

动作框里常用与互锁相关的指令如表 6-2 所示。

表 6-2　动作框里常用与互锁相关的指令

指令(符号)	指令基本动作描述
NC	当该“步”为活动步并且互锁条件满足时地址输出“1”;当该“步”为不活动步时地址输出“0”
SC	当该“步”为活动步并且互锁条件满足时地址输出“1”并保持(即置位)
RC	当该“步”为活动步并且互锁条件满足时地址输出“0”并保持(即复位)
DC	当该“步”为活动步并且互锁条件满足时,开始计时(时间由该框 T#xx 指定),当时间到地址输出“1”; 当该“步”为不活动步时地址输出“0”
LC	当该“步”为活动步并且互锁条件满足时,地址输出“1”并开始计时(时间由该框 T#xx 指定),当时间到地址输出“0”; 当该“步”为不活动步时地址输出“0”
CALLC	当该“步”为活动步并且互锁条件满足时,调用指定的程序块

例如:“S1 ON C　S005”的意思是当步 1（S1）激活瞬间并且互锁条件满足，则激活步 5（即 Sequencer 2 中的 S5）。

像上面这种与事件相关的指令称为事件指令，常用的事件指令如图 6-40 所示。动作框里常用的事件指令如表 6-3 所示。

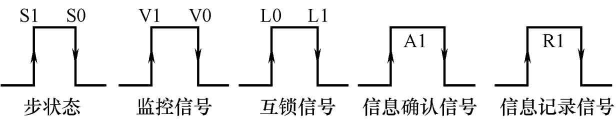


图 6-40　常用的事件指令

表 6-3　动作框里常用的事件指令

指令(符号)	相关的事件描述
S1	该步激活瞬间
S0	该步取消激活瞬间
V1	该步发生监控错误瞬间
V0	该步监控错误消除瞬间
A1	确认错误信息信号
R1	在“REG_EF”或“REG_S”的输入信号,即是记录信息信号

动作框里常用与事件相关的指令除了“D”和“L”以外，其他的常用指令都可以进行组合使用。可以与事件指令组合使用的常用指令如表 6-4 所示。

在 FB1 的 Sequencer 1 里编写程序，如图 6-41 所示。在 OB1 里编写的程序如图 6-42 所示。

表 6-4　与事件指令组合使用的常用指令（图中以“S1”步激活事件为例）

指令(符号)		解　　说
S1 N(C)	Q0.0	在该步激活瞬间(且互锁条件满足),Q0.0 接通一个周期
S1 S(C)	M10.0	在该步激活瞬间(且互锁条件满足),置位 M10.0

（续）

指令(符号)		解　　说
S1 R(C)	M12.1	在该步激活瞬间(且互锁条件满足),复位 M12.1
S1 CALL(C)	FB20	在该步激活瞬间(且互锁条件满足),调用 FB20
S1 ON(C)	S7	在该步激活瞬间(且互锁条件满足),激活步 7(S7)
S1 OFF(C)	S5	在该步激活瞬间(且互锁条件满足),取消激活步 5(S5)
	S_ALL	在该步激活瞬间(且互锁条件满足),取消激活所有步(除该指令的步外)
S1 SC(C)	C10	在该步激活瞬间(且互锁条件满足),装载 C10 的计数值为 12(范围:0 ~ 999)
	C#12	
S1 CU(C)	C13	在该步激活瞬间(且互锁条件满足),C13 加 1(范围:0 ~ 999)
S1 CD(C)	C14	在该步激活瞬间(且互锁条件满足),C14 减 1(范围:0 ~ 999)
S1 CR(C)	C15	在该步激活瞬间(且互锁条件满足),复位 C15
S1 TD(C)	T10	在该步激活瞬间(且互锁条件满足),启动 T10 定时器,定时事件到,触点 T10 为 1;启动后定时器开始计时并且与互锁条件和该步的状态无关,具有闭锁功能;设定值可以使用变量指定,也可以直接使用 S5 的时间值指定
	S5T#12S	
S1 TL(C)	T12	在该步激活瞬间(且互锁条件满足),启动 T12 以扩展脉冲方式定时,没有闭锁功能;设定值可以使用变量指定,也可以直接使用 S5 的时间值指定
	MW15	
S1 TR(C)	T20	在该步激活瞬间(且互锁条件满足),复位定时器 T20
S1 N(C)	MW10: = MW14	在该步激活瞬间(且互锁条件满足),把 MW14 的值赋予 MW10;数据类型可以是 8 位、16 位及 32 位
S1 N(C)	MW10: = MW12 + IMW14	在该步激活瞬间(且互锁条件满足),把 MW12 + MW14 的和赋予 MW10;可以使用常用的运算符号
S1 N(C)	A: = 函数(B)	在该步激活瞬间(且互锁条件满足),把 B 按照指定函数运算的结果赋予 A;函数使用 S7 Graph 内置的函数

6.2.5　监控程序的应用

监控程序就像它的名字一样，是监控编程的程序正常运行的程序，可能读者会想为什么要监控程序的正常运行呢?

实际上在机械控制系统中，安全系数是非常重要的。尽管编写的程序没有错误，自动挡可以正常运行，但是在单步或手动挡调试的时候问题就出现了。解决这些问题可以使用互锁和监控的辅助功能编写限制条件程序。

我们以平台左右运动控制系统为例进行介绍，平台左右运行示意图如图 6-43 所示。

在正常情况下平台只能在 B ~ C 之间运动，A 和 D 点分别是左边和右边的限位开关，当限位开关接通时表示有问题产生，当从右往左运动时间超出 10s 也表示有问题产生，当从左往右运动时间超出 8s 也表示有问题产生。

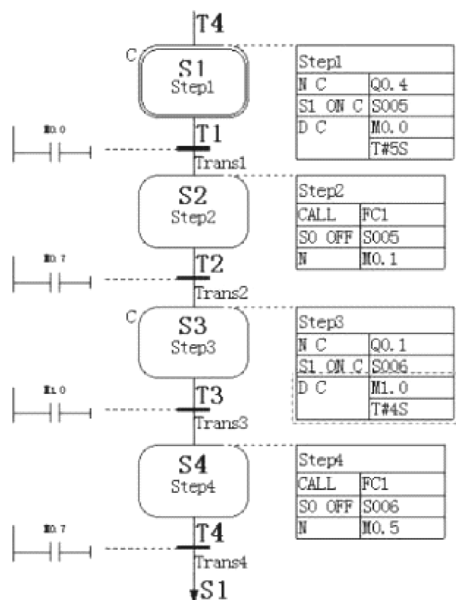


图 6-41　在 FB1 的 Sequencer1 里编写的程序

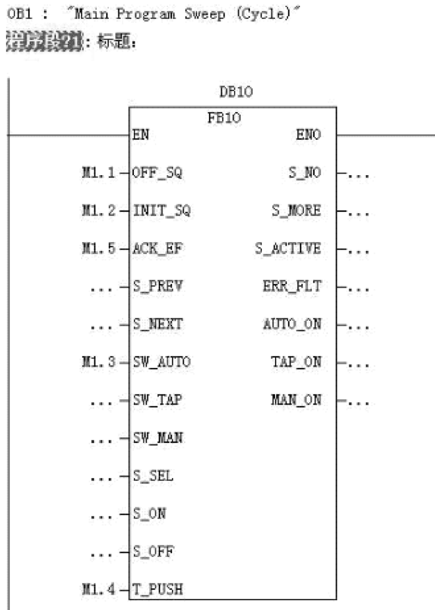


图 6-42　在 OB1 里编写的程序

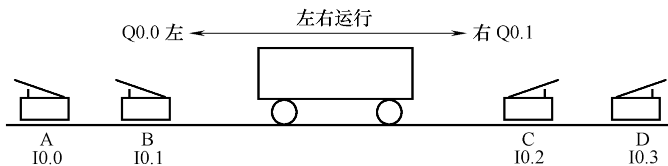


图 6-43　平台左右运行

Q0.0 =1 平台往左运动，I0.1 是左边的到位开关，I0.0 是左边的限位开关（极限开关）。

Q0.1 =1 平台往右运动，I0.2 是右边的到位开关，I0.3 是右边的限位开关（极限开关）。在 I0.1 =0 和 Q0.1 =0 的前提下，才有可能往左运动；在 I0.2 =0 和 Q0.0 =0 的前提下，才有可能往右运动。

1）通过上面的解释，我们编写控制往左运动的控制程序如图 6-44 所示。

2）控制往右运动的控制程序如图 6-45 所示，在 OB1 里编写的程序如图 6-46 所示。

图 6-44 和图 6-45 中的“Supervision”（监控）里分别出现“S001. T”和“S002. U”。其中：

“S001. T”的意思是：

1）当步 1 在激活时累计步 1 本次激活的时间；

2）当步 1 不激活时，表示上次激活的总时间。可以使用“Sx. T”表示任意步激活的总时间，“x”是步的地址编号，以下出现的具有相同的意思。

“S002. U”的意思是：

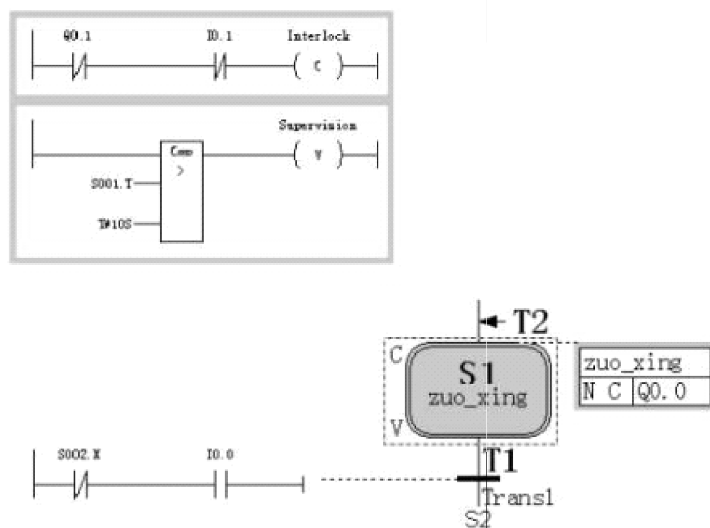


图 6-44　往左运动的控制程序

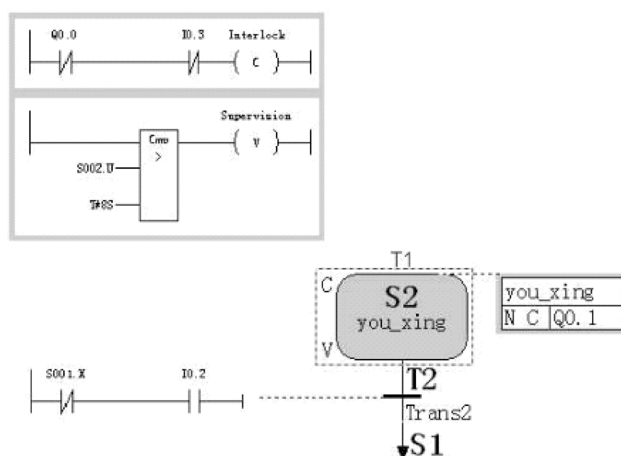


图 6-45　往右运动的控制程序

1）当步 1 在激活时累计步 1 本次激活的时间，这个时间是不含有干扰的时间；

2）当步 1 不激活时，表示上次激活的时间，这个时间是不含有干扰的时间。可以使用“Sx. U”表示任意步激活的时间。

干扰事件包括互锁不满足或超出监控设置的时间。干扰事件发生的过程时间称为干扰时间。为了区分“Sx. T”与“Sx. U”，可以把 Sx. T 称为总时间，Sx. U 称为有效时间。

在图 6-44 和图 6-45 中的转移条件里分别出现“S002. X”和“S001. X”，其 中：

当 S002. X =0 时，表示步 2 没有激活，当 S002. X =1 时，表示步 2 激活了。

当 S001. X =0 时，表示步 1 没有激活，当 S001. X =1 时，表示步 1 激活了。

像“S001. X”使用特定“字符 + 地址”来表达“步 1”的系统状态，称为系统信息地址。常用的系统信息“地址”如表 6-5 所示，这些地址可以在转换、监控、互锁和永久指的地方使用。

表 6-5　以地址表示特定的系统信息

地址	使用场合	特例	解说
Sx. T	比较指令	S1. T	步 1 激活的总时间
Sx. U		S2. U	步 2 激活的有效时间
Sx. X	触点指令	S3. X	步 3 激活时 S3. X = 1；步 3 不激活时 S3. X = 0
Transx. TT		Trans 4. TT	步 4 转移条件满足时 Trans 4. TT = 1； 步 4 转移条件不满足时 Trans 4. TT = 0

6.2.6　功能块输入和输出参数

在 S7 Graph 编辑界面，单击“Options”→“Block Settings”→“Compile/Save”，可以选择 S7Graph FB 的参数模式，如图 6-47 所示，其中有 4 种参数模式。

1）Minimum：最小参数模式，只有自动控制模式，没有监视功能。最小参数模式功能块图如图 6-48a 所示。

2）Standard：标准参数模式，有多种控制模式及状态信息供选择，可以选择信息记录及确认。标准参数模式功能块图如图 6-48b 所示。

3）Maximum（≤4）：V4 及以下版本的最大参数模式，可以提供更多的操作员控制、调试和监控功能。V4 及以下版本参数模式功能块图如图 6-48c 所示。

4）User-defined（V5. x）：V5 版本的最大参数模式，可以提供更多的操作员控制、调试和监控功能。V5 版本参数模式功能块图如图 6-48d 所示。

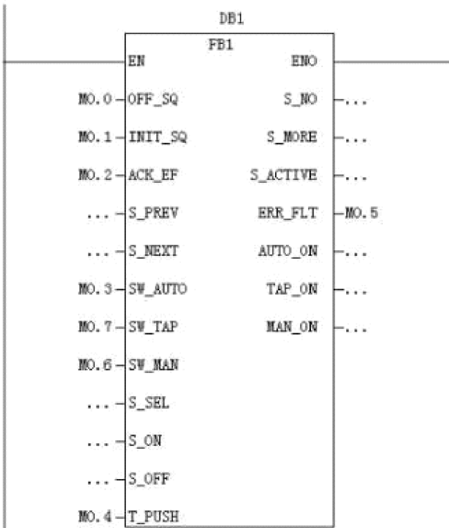


图 6-46　在 OB1 里编写的程序

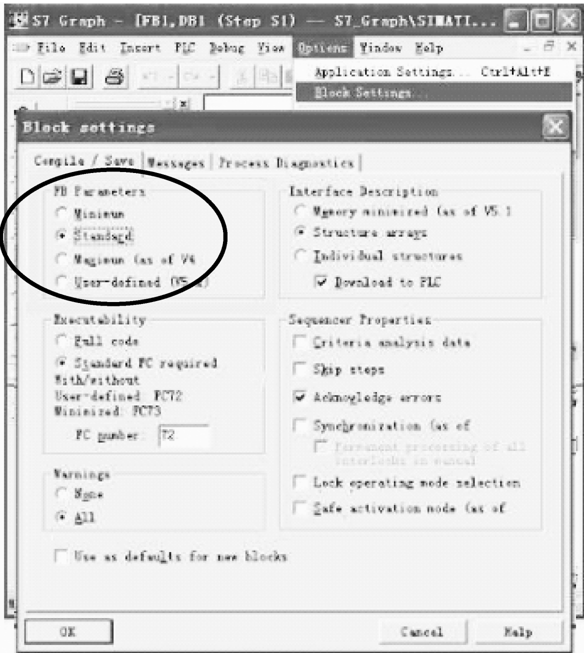


图 6-47　功能块参数设置界面

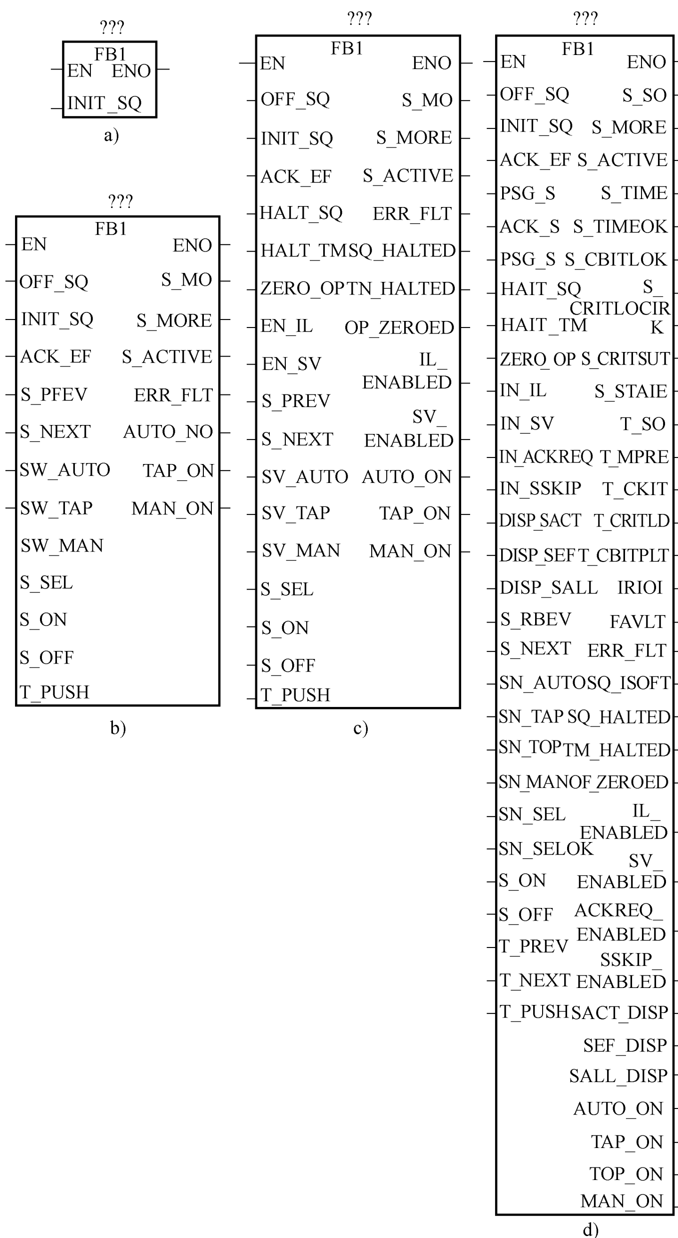


图 6-48　4 种参数模式的功能块

第 7 章

S7-300/400的网络通信

7.1 西门子 PLC 网络

7.1.1 现代工业企业典型网络结构

自动化系统一般是以单元生产设备为中心进行检测和控制，不同单元的生产设备间缺乏信息交流，难以满足生产过程的统一管理。在市场竞争日益激烈的今天，企业里配备有不同单元的生产设备，而希望它们之间能够互相交流信息；不同厂家生产的设备能够互相配合，实现生产现场级与管理级的信息共享，提高效率，降低成本，从而提高自身在市场上的竞争力。为了提高整个企业在市场上的竞争力，实现最佳经济效益，必须将自动化控制、制造执行系统（MES）和企业资源规划系统（ERP）互相协调，形成一个整体。

基于以上需求，不同网络体系结构的用户需要一套通用的计算机网络通信标准，于是国际相关组织提出了网络的基本参考模型（OSI/RM）。

西门子全集成自动化解决方案顺应了当今自动化的需求，TIA 将统一的组态和编程、统一的数据管理及统一的通信 3 方面集成在一起。从现场级到管理级，TIA 通信覆盖了整个企业，其优越的通信网络适应各种应用，如工业以太网、PROFIBUS、MPI、AS-i 总线及 EIB 总线（PROFIBUS 总线的扩展）。

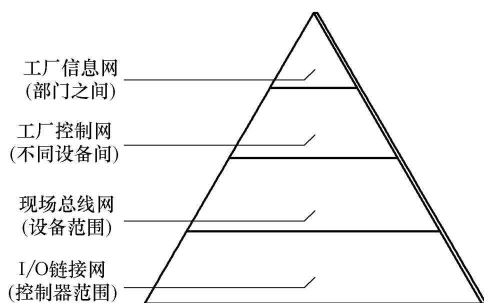


图 7-1　工厂自动化网的构成

工业化网络系统需要根据企业的自动化程度与要求分层次构建。按网络的控制范围、要求与功能，工厂自动化网一般分为图 7-1 所示的 4 个层次，并按金字塔形式布置。

1. I/O 链接网

I/O 链接网是设备控制器范围内的网络层。当控制器为 PLC 时，I/O 链接网是指 PLC 与检测元件、执行元件等 I/O 装置连接的网络层，它是工厂自动化网的最底层，其使用最为广泛。

I/O 链接网可将远离控制器但又相对集中的各种执行元件、检测元件汇总连接到远程 I/O 站上，然后通过简单的总线（一般为 2 芯电缆）与控制器进行通信连接，实现对远程 I/O

点的控制。使用 I/O 链接网可大大节省执行元件、检测元件与控制器的连接线，故又称“省配线网”。

I/O 链接网通常采用执行器—传感器总线接口（Actuator Sensor Interface，简称 AS-i 网）、I/O 链接总线接口（I/O-Link，简称 I/O-Link 网）进行链接。

2. 现场总线网

现场总线（Field Bus）用来连接设备上各种不同控制装置的多点通信串行数据总线（Multiple Points Interface，MPI）。现场总线可使得设备上的 PLC、数控机床（CNC）、驱动器等控制装置构成一个完整的控制系统，以便进行集中与统一的管理。

3. 工厂控制网

工厂控制网是连接生产车间中不同设备的网络系统，它可将车间内部的各种加工、检测设备连接成一个完整的网络系统，以便进行不同控制设备间的数据实时传送与集中管理。

4. 工厂信息网

工厂信息网是整个工厂的生产管理局域网，多采用开放型的工业以太网（Ethernet）进行连接。工厂信息网可对工厂各生产现场的数据进行收集、整理，并对生产计划进行统一的管理与调度。

7.1.2　西门子 PLC 网络分类

西门子 PLC 网络组合方式多种多样，常见的有以下几种：

1. 自由口通信

自由口通信也称用户自定义协议通信，它主要针对 S7-200 系列的 PLC。自由端口模式下，用户可通过发送指令（XMT）、接收指令（RCV）、发送中断、接收中断等来控制通信口的操作。

一般情况下，第三方设备大都支持 RS-485 串口通信，西门子 S7-200 PLC 可以通过选择自由口通信模式控制串口通信；同时，自由口通信也为计算机与 S7-200 PLC 之间的通信提供了一种廉价与灵活的方法。计算机与 PLC 通信时，为了避免各方争用信道，一般采用主从方式，即计算机为主机，PLC 为从机，只有主机才有权主动发送请求报文，从机收到后返回响应报文。自由口通信也可以用于 PLC 之间的通信。

需要注意的是：

自由口通信模式下，计算机与 S7-200 PLC 之间通信是指上位机编程软件 STEP 7 与 S7-200 CPU 之间的通信，通信协议完全由梯形图程序控制。

---

2. PPI（Point to Point 点对点接口）通信协议

PPI 通信协议是西门子公司专门为 S7-200 系列 PLC 开发的一种通信协议，也是 S7-200 PLC 的默认通信协议，一般不对外开放。PPI 协议是 S7-200 CPU 最基本的通信方式，通过原来自身的端口（PORT0 或 PORT1）就可以实现通信。

PPI 是一种主—从协议通信，主—从站在一个令牌环网中。在 CPU 内用户网络即可读写指令，也就是说网络读写指令是运行在 PPI 协议上的，因此 PPI 只在主站侧编写程序就可以了，从站的网络读写指令没有什么意义。网络上的 S7-200 CPU 均为从站，其他 CPU、SI-

MATIC 编程器或 TD200 文本显示器为主站。

3. MPI（Multi Point Interface 多点接口）**通信协议**

MPI 通信是当通信速率要求不高，通信数据量不大时可以采用的一种简单经济的通信方式。通过它可组成小型 PLC 通信网络，实现 PLC 之间的少量数据交换，它不需要额外的硬件和软件就可网络化。每个 S7-300 CPU 都集成了 MPI 通信协议，MPI 的物理层是 RS-485。

通过 MPI，PLC 可以同时与多个设备建立通信连接，这些设备包括编程器 PG 或运行 STEP 7 的计算机、人机界面（HMI）及其他 SIMATIC S7、M7 和 C7。同时连接的通信对象的个数与 CPU 的型号有关。MPI 通信速率为 9.2kbit/s ~ 12Mbit/s，其默认的速率为 187.5kbit/s，MPI 网络最多可连接 32 个节点，最大通信距离为 50m，可以用 RS-485 中继器来扩展其通信范围。

4. PROFIBUS（Process Field Bus 过程现场总线）**通信协议**

PROFIBUS 通信是一种用于工厂自动化车间级监控和现场设备层数据通信与控制的现场总线技术，可实现现场设备层到车间级监控的分散式数字控制和现场通信网络，从而为实现工厂综合自动化和现场设备智能化提供了可行的解决方案。PROFIBUS 是一种开放式总线标准，是不依赖于设备生产商的现场总线标准。传输速率可在 9.6kbit/s ~ 12Mbit/s 间选择。



注意：

如何通过 PROFIBUS-DP 用功能块实现在主、从站之间双向数据传送？

在主站 PLC 可以通过调用 SFC14 “DPRD_ DAT” 和 SFC15 “DPWR_ DAT” 来完成和从站的数据交换，而对于从站来说可以调用 FC1 “DP_ SEND” 和 FC2 “DP_ RECV” 完成数据的交换。

PROFIBUS 连接的系统由主站和从站组成，主站和从站可以是一个，也可以是多个。主站能够控制总线，多主站时通过令牌的传递来决定哪个主站享有控制权，从站一般为传感器、变送器、驱动器等。PROFIBUS 是在欧洲工业界得到应用的一个现场总线标准。

PROFIBUS 通信由 3 个兼容部分组成，即 PROFIBUS-DP（Decentralized Periphery）车间级通信、PROFIBUS-PA（Process Automation）现场级通信、PROFIBUS-FMS（Fieldbus Message Specification）工厂级通信。PROFIBUS-DP 是一种高速低成本通信，用于设备级控制系统与分散式 I/O 的通信。使用 PROFIBUS-DP 可取代 DC24V 或 4 ~ 20mA 信号传输。PORFIBUS-PA 专为过程自动化设计，可使传感器和执行机构连在一根总线上，并有本征安全规范；PROFIBUS-FMS 用于车间级监控网络，是一个令牌结构，实时多主网络。这 3 个部分用的协议也不相同。如图 7-2 所示，每一级可采用不同的通信方式。

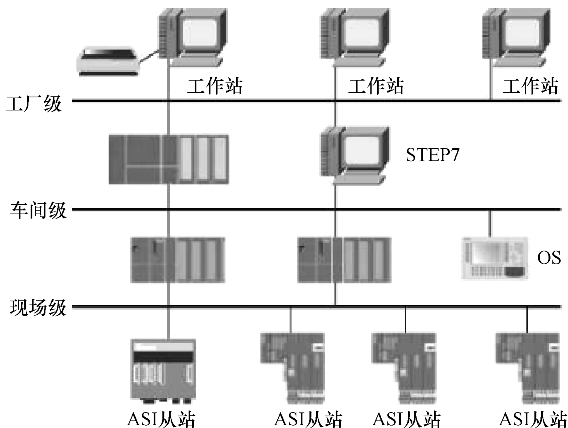


图 7-2 PROFIBUS 的组成

5. 工业以太网（Industrial Ethernet）通信

工业以太网技术是普通以太网技术在控制网络延伸的产物，以太网技术经过多年的发展，特别是它在 Internet 中的广泛应用，使其技术更为成熟。随着技术的发展，控制网络与普通计算机网络、Internet 的联系更为密切。控制网络技术需要考虑与计算机网络连接的一致性，这些都使得工业以太网较之其他类型网络技术具有明显的优势。

工业以太网与普通以太网相比，存在着一些不同之处，如应用场合、拓扑结构与网络监控等。工业以太网一般技术上与普通以太网（即 IEEE 802.3 标准）兼容，但在产品设计时，在材质的选用、产品的强度、适用性以及实时性、可互操作性、可靠性、抗干扰性和本质安全等方面要能满足工业现场的需要。

西门子公司在工业以太网领域有着非常丰富的经验和领先的解决方案。其中 SIMATIC NET 工业以太网基于经过现场验证的技术，符合 IEEE 802.3 标准并提供 10～100Mbit/s 的快速以太网技术，为用户提供了开放的、适用于工业环境下各种控制级别的不同控制系统。西门子公司为工业以太网通信提供了全方位的产品支持，S7 系列 PLC 常用工业以太网通信处理器（Communicaton Processer，CP）提供对工业以太网的通信支持。

7.2　MPI 通信概述

7.2.1　MPI 网络概述

MPI 通信是 S7-300/400 CPU 默认的一种通信方式，也是一种比较简单的通信方式，MPI 网络通信的速率是 19.2kbit/s～12Mbit/s，MPI 网络最多支持连接 32 个节点，最大通信距离为 50m，可以通过中继器扩展通信距离，但中继器也占用节点。

西门子 PLC　S7-200/300/400　CPU 上的 RS-485 接口不仅是编程接口，同时也是一个 MPI 的通信接口，不增加任何硬件就可以实现 PG/OP、全局数据通信及少量数据交换的 S7 通信等功能。MPI 网络节点通常可以挂 S7 PLC、人机界面、编程设备、智能型 ET200S 及 RS-485 中继器等网络元器件。通常的网络结构配置如图 7-3 所示。

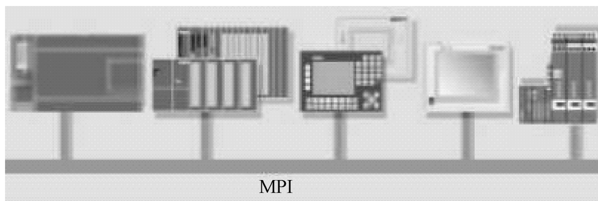


图 7-3　西门子 MPI 通信网络配置图


注意：

为什么不能通过 MPI 在线访问 CPU?

如果在 CPU 上已经更改了 MPI 参数，请检查硬件配置。可以将这些值与在“SetPG/PCinterface”下的参数进行比较，看是否有不一致。

或者可以这样做：打开一个新的项目，创建一个新的硬件组态。在 CPU 的 MPI 属性中为地址和传送速度设置各自的值。将“空”项目写入存储卡中。把该存储卡插入到 CPU 然后重新打开 CPU 的电压，将位于存储卡上的设置传送到 CPU。现在已经传送了 MPI 的当前设置，并且像这样的话，只要接口没有故障就可以建立连接。这个方法适用于所有具有存储卡接口的 S7-CPU。

7.2.2　通信方式

西门子 PLC 与 PLC 之间的 MPI 通信一般有 3 种通信方式：全局数据包通信方式、无组态连接通信方式和组态连接通信方式。

（1）全局数据包通信方式

全局数据包通信（GD 通信）是集成在 S7-300 和 S7-400　CPU 操作系统中的一种简单通信方式。GD 通信将允许通过多点接口在 CPU 之间对数据进行循环交换。循环数据交换将随正常的过程映像产生。

GD 通信只能在 S7-300 与 S7-300、S7-400 与 S7-400 或 S7-300 与 S7-400 之间进行，用户不需要编写任何程序，在硬件组态时组态所有 MPI 通信的 PLC 站间的发送区与接收区就可以了。

（2）无组态连接通信方式

无组态连接的 MPI 通信适合在 S7-300 与 S7-400、S7-200 之间进行，调用 SFC65、SFC66、SFC67、SFC68 或 SFC69 来实现。值得注意的是，无组态连接通信方式不能与全局数据包通信方式混合使用。无组态连接的 MPI 通信又分两种：双边编程通信方式与单边编程通信方式。

双边编程通信方式：就是本地与远程两方都要编写通信程序，发送方使用 SFC65 来发送数据，接收方用 SFC66 来接收数据，这些系统功能只有 S7-300/400 才有，因此双边编程通信方式只能在 S7-300/400 之间进行，不能与 S7-200 通信。

单边编程通信方式：只在一方编写程序，好像客户机与服务器的访问模式，编写程序一方就像是客户机，不编写程序一方就像是服务器。这种通信方式符合 S7-200 与 S7-300/400 之间的通信，如果是 S7-200 CPU 那就只能做服务器。使用 SFC67 系统功能来读取对方指定的地址数据到本地机指定的地方存放，使用 SFC68 系统功能来将本地机指定的数据发送到对方指定的地址区域存放。

（3）组态连接通信方式

如果交换的信息量较大时，可以选择组态连接通信方式，这种通信方式只能在 S7-300 与 S7-400 或 S7-400 与 S7-400 之间进行。在 S7-300 与 S7-400 之间通信时，S7-300 只能做服务器，S7-400 只能做客户机；在 S7-400 与 S7-400 之间进行通信时，任意一个 CPU 都可以做服务器或客户机。

下面逐一介绍这 3 种 MPI 通信的组态和编程方法及步骤。

7.2.3　基本 MPI 的项目组态实例

1. 全局数据包通信方式

1）硬件与软件要求：所需硬件有 CPU 315-2 DP 和 CPU 414-2 DP；所需软件有 STEP 7 V5.3。

2）网络配置：网络配置如图 7-4 所示。

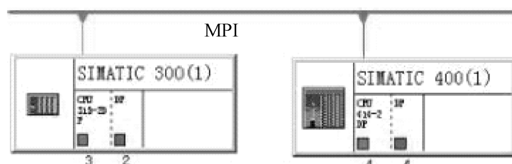


图 7-4　GD 通信网络配置

3）新建一个项目“GD 通信”，分别

插入两个站 S7-300 和 S7-400，并完成电源和 CPU 的硬件组态，如图 7-5 所示。

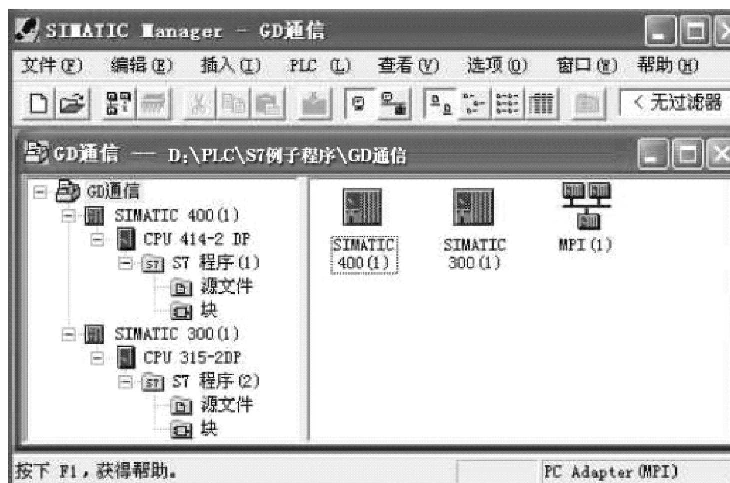


图 7-5　新建名为“GD 通信”的项目

4）首先组态 S7-400 站的 MPI 网。在 S7-400 硬件组态界面，双击机架上的 CPU 414-2DP，将弹出 CPU 属性界面，如图 7-6 所示。

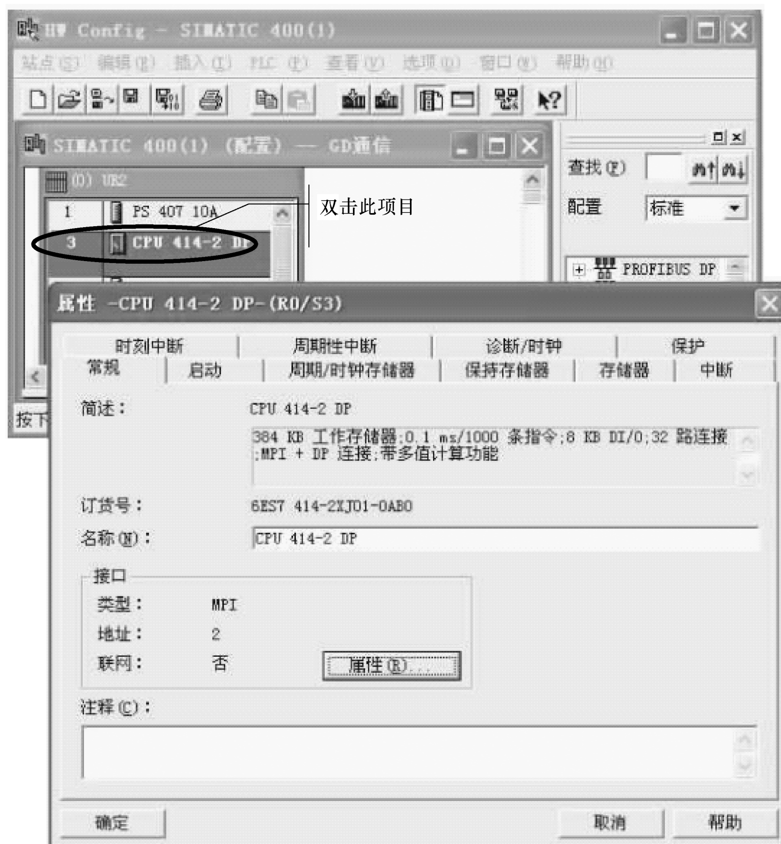


图 7-6　CPU 属性界面

在 CPU 属性界面里单击“属性”按钮，将弹出 MPI 属性界面，如图 7-7 所示，设置 S7-400 的 MPI（1）地址为 4，通信波特率为 187.5kbit/s。



图 7-7　MPI 属性界面

然后利用同样方法组态 S7-300 站的 MPI 网。

在 S7-300 硬件组态界面，双击机架上的 CPU 315-2DP，将弹出 CPU 属性界面，在 CPU 属性界面里单击“属性”按钮，将弹出 MPI 属性界面，设置 S7-300 的 MPI（1）地址为 3，通信波特率为 187.5kbit/s。

5）接下来需要打开“GD-MPI（1）”表格 。在管理画面，单击项目名称“GD 通信”，在右边出现“MPI（1）”图标，双击该图标，如图 7-8 所示，然后选中 MPI（1）红色总线，单击“选项”→“定义全局数据”，将弹出设置“GD-MPI（1）”表格。



图 7-8　打开“GD-MPI（1）”表格

6）在“GD-MPI（1）”表格界面，双击图 7-9 所示的“A”处，在弹出选择 CPU 的界面里选择“CPU 414-2DP”，然后单击“确定”。



图7-9　在“GD-MPI（1）”表格插入CPU

利用同样方法在图7-9的“B”处插入“CPU 315-2DP”。

7）在“GD-MPI（1）”表格界面，在1行与CPU414-2DP交叉的表格里写上“MB10：10”，其中MB10：10后面的“10”是接收或发送的字节个数，MB10是接收或发送的开始字节。S7-300接收或发送的最多字节数是22个，S7-400则是54个，发送区与接收区要求相同。接收和发送的地址区可以是DB、M、I、Q。单击“MB10：10”，按动右键选择下拉的“发送器”，表示发送MB10～MB19这10个字节的数据，如图7-10所示。



图7-10　声明CPU414-2DP的发送区

利用同样方法声明MB130开始连续20个字节为发送区，如图7-11所示。

8）最后单击“ ”，编译保存组态MPI接口的数据区数据。CPU315-2DP与CPU414-2DPMPI（1）的接口数据对应关系如表7-1所示。

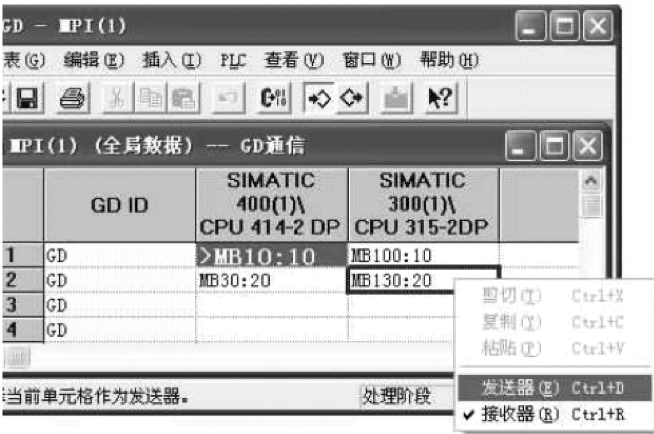


图 7-11　声明 CPU315-2DP 的发送区

表 7-1　MPI（1）的接口数据对应关系

CPU315-2DP(3 号站)	对应关系	CPU414-2DP(4 号站)
MB100 ~ MB109	←————	MB10 ~ MB19
MB130 ~ MB149	————→	MB30 ~ MB49

9）组态完 MPI 全局数据包通信发送和接收数据，在编译后，就看到在每行通信区都有 GD IN 号，如图 7-12 所示。

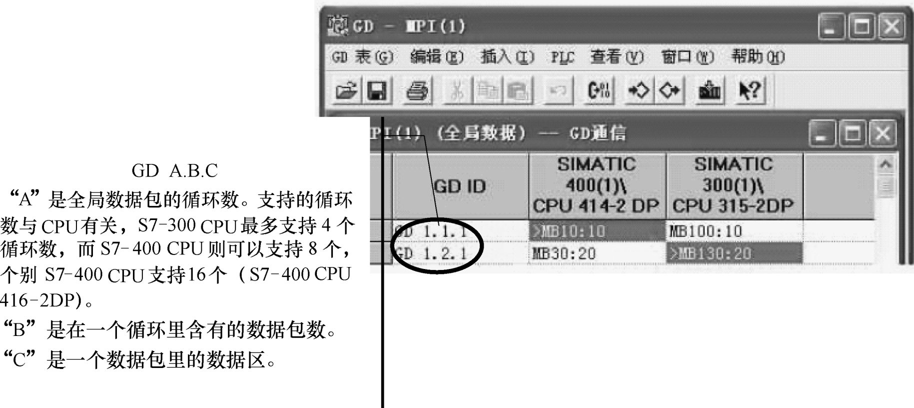


图 7-12　组态参数的含义

图 7-12 中 A、B、C 参数是为了优化接收与发送区，在一般的应用中可以不用理会，GD IN 号在编译后自动生成。

10）在通信时，可能出现通信中断，可以按照以下办法检测。如图 7-13 所示单击“查看”，在下拉菜单中分别选择“扫描速率”和“GD 状态”，可以查看扫描系数和状态字，如图 7-14 所示。

在图 7-14 中“SR1. 1”为 23，表示发送更新时间为 23 × CPU 循环扫描时间，SR 范围为 1 ~ 255。如果 SR 值设置太小，容易出现通信中断，可以按照具体需要设置大一点。GST 是所有 GDS 逻辑或运算的结果。图 7-14 中 GDS 每包数据的状态字的各位意义如表 7-2 所示。

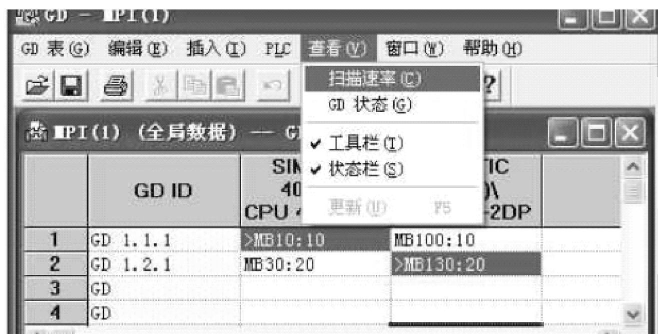


图 7-13　打开扫描系数和状态字

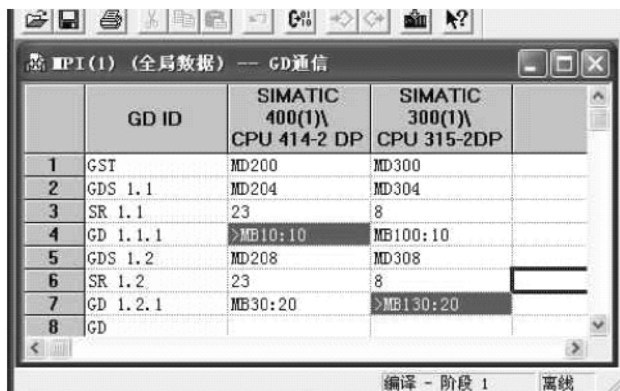


图 7-14　扫描系数和状态字

表 7-2　GDS 状态字的意义

Bit	表示意义	Bit	表示意义
1	发送区长度错误	7	发送区与接收区长度不一致
2	发送区数据块不存在	8	接收区长度错误
4	全局数据包丢失	9	接收区数据块不存在
5	全局数据包语法错误	12	发送方重新发送
6	全局数据包数据对象丢失	32	接收区重新接收

11）在全局数据包通信时如果需要控制数据的发送与接收，需要调用系统功能 SFC60 及 SFC61，支持这种功能的只有 S7-400 CPU，把需要控制发送与接收的 SR 改为 0 就可以了，如图 7-15 所示。

通过 SFC61 “GD RCV”（全局数据接收），从进入的 GD 帧中为单个 GD 信息包提取数据，然后输入接收到的 GD 信息包中。SFC61 “GD RCV” 的输入/输出参数如表 7-3 所示。

通过 SFC60 “GD SND”（全局数据发送），采集 GD 信息包的数据，并通过在 GD 信息包中指定的路径发送。SFC60 “GD SND” 的输入/输出参数如表 7-4 所示。

在图 7-16 程序中，当 M0.0 接通时，S7-400 就接收 S7-300 的数据，将 S7-300 的 MB130 ~ MB149 中的数据接收到 S7-400 的 MB30 ~ MB49 中；当 M0.2 接通时，S7-400 发送数据，将 S7-400 的 MB10 ~ MB19 的数据发送到 S7-300 中的 MB100 ~ MB109 中。

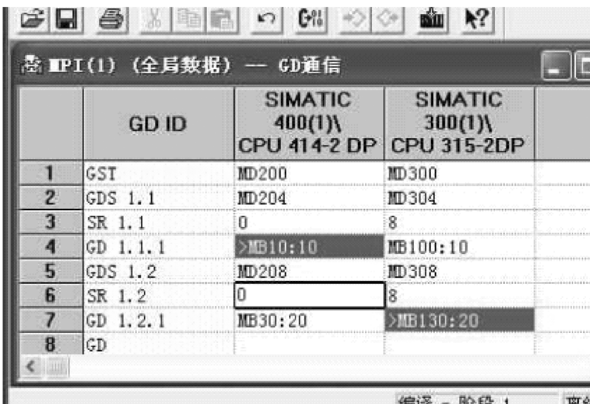


图 7-15　把 S7-400CPU 的 SR 改为“0”

表 7-3　SFC61“GD RCV”的输入/输出参数

参数符号	参数类型	数据类型	说　　明
CIRCLE_ID	INPUT	BYTE	用于输入进入 GD 信息包的 GD 环的数目，允许使用的数值：1～16
BLOCK_ID	INPUT	BYTE	所选择的 GD 环中的 GD 信息包的数目，将在其中输入进入的数据，允许使用的数值：1～3
RET_VAL	OUTPUT	INT	错误信息

表 7-4　SFC60“GD SND”的输入/输出参数

参数符号	参数类型	数据类型	说　　明
CIRCLE_ID	INPUT	BYTE	要发送的 GD 信息包所在的 GD 环的数目，允许使用的数值：1～16
BLOCK_ID	INPUT	BYTE	要在所选择的 GD 环中发送的 GD 信息包的数目，允许使用的数值：1～3
RET_VAL	OUTPUT	INT	错误信息

2. 无组态连接通信方式

无组态连接通信方式可以不依赖组态 GD 通信包，而是使用用户编程方式实现 MPI 通信功能。用户编程时灵活使用 SFC65～SFC69 来组织程序。

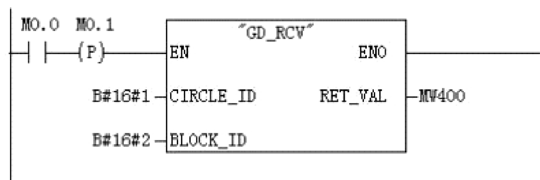
通过 SFC65“X SEND”发送数据到 MPI 网络指定的通信伙伴。在通信伙伴上使用 SFC66“X RCV”来接收数据。SFC65“X—SEND”和 SFC66“X RCV”符合双边编程的 MPI 通信。

SFC65“X—SEND”的输入/输出参数如表 7-5 所示；SFC66“X—SEND”的输入/输出参数如表 7-6 所示。

通过 SFC67“X GET”，可以读取 MPI 网上通信伙伴中的数据，在通信伙伴上没有相应的发送程序。SFC67“X GET”指令符合单边编程的 MPI 通信。SFC67“X GET”的输入/输出参数如表 7-7 所示。

OB1 : "Main Program Sweep (Cycle)"主站程序

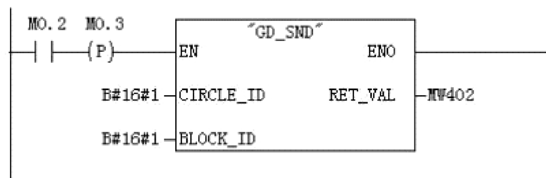
程序段?1：标题：



符号信息：

GD_RCV　　SFC61　-- Fetch a Received GD Packet

程序段?2：标题：



符号信息：

GD_SND　　SFC60　-- Send a GD Packet

图 7-16　在 S7-400PLC 的 OB1 里编写的控制程序

表 7-5　SFC65“X—SEND”的输入/输出参数

参数符号	参数类型	数据类型	参 数 解 说
REQ	INPUT	BOOL	=1 时表示发送请求，建立通信动态连接
CONT	INPUT	BOOL	=1 时表示发送的数据是一个整体不能分割
DEST_ID	INPUT	WORD	发送到对方(通信伙伴)的 MPI 地址
REQ_ID	INPUT	DWORD	作业标志符，此参数用于识别通信伙伴上的数据
SD	INPUT	ANY	发送数据区，比如 P#M20.0 BYTE 5，表示把本地 MB20 开始的 5 个字节数据发送出去，最大发送字节数为 76 个字节
RET_VAL	OUTPUT	INT	如果在该功能执行过程中出错，则返回值包含相应的错误代码
BUSY	OUTPUT	BOOL	=1 表示正在发送，=0 表示发送完成或不存在

表 7-6　SFC66“X—SEND”的输入/输出参数

参数符号	参数类型	数据类型	参 数 解 说
EN_DT	INPUT	BOOL	=1 表示接收使能，“激活数据传送”
RET_VAL	OUTPUT	INT	如果在该功能执行过程中出错，则返回值包含相应的错误代码
REQ_ID	OUTPUT	DWORD	作业标志符，它的数据位于队列的开头，即队列中最早的数据，如果队列中没有数据块，则 REQ_ID 数值为 0
NDA	OUTPUT	BOOL	=1 表示有新发来的信息，=0 则没有
RD	OUTPUT	ANY	指向接收数据区域，最大接收字节数是 76 个字节

表 7-7 SFC67“X GET”的输入/输出参数

参数符号	参数类型	数据类型	参 数 解 说
REQ	INPUT	BOOL	=1 时表示接收请求,建立通信动态连接
CONT	INPUT	BOOL	=1 时表示接收的数据是一个整体不能分割
DEST_ID	INPUT	WORD	对方(通信伙伴)的 MPI 地址
VAR_ADDR	INPUT	ANY	指向伙伴 CPU 上要从中读取数据的区域,必须选择通信伙伴支持的数据类型
RET_VAL	OUTPUT	INT	如果在该功能执行过程中出错,则返回值包含相应的错误代码
BUSY	OUTPUT	BOOL	=1:接收还没有结束。=0:接收已经结束,或当前没有激活的接收作业
RD	OUTPUT	ANY	指向接收数据区,接收区的最大长度是 76 个字节

通过 SFC68“X PUT”，将数据发送到 MPI 网上通信伙伴中的指定数据区，在通信伙伴上没有相应接收程序。SFC68“X PUT”指令符合单边编程的 MPI 通信。SFC68“X PUT”的输入输出参数如表 7-8 所示。

表 7-8 SFC68“X PUT”的输入/输出参数

参数符号	参数类型	数据类型	参 数 解 说
REQ	INPUT	BOOL	=1 时表示发送请求,建立通信动态连接
CONT	INPUT	BOOL	=1 时表示发送的数据是一个整体不能分割
DEST_ID	INPUT	WORD	对方(通信伙伴)的 MPI 地址
VAR_ADDR	INPUT	ANY	指向伙伴 CPU 中要写入数据的区域,必须选择通信伙伴支持的数据类型
SD	INPUT	ANY	指向本地 CPU 中包含要发送数据的区域,发送区的最大长度是 76 个字节
RET_VAL	OUTPUT	INT	如果在该功能执行过程中出错,则返回值包含相应的错误代码
BUSY	OUTPUT	BOOL	=1:发送还没有结束。=0:发送已结束,或当前没有激活的发送作业

通过 SFC69“X ABORT”终止通过 SFCX SEND、X GET 或 X PUT 建立的通信动态连接。

如果属于 X SEND、X GET 或 X PUT 的作业已结束（BUSY =0），则在调用 SFC69“X—ABORT”之后，将释放在通信两端使用的连接资源。

如果属于 X SEND、X GET 或 X PUT 的作业还没有结束（BUSY =1），则在连接终止之后重新通过 REQ =0 和 CONT =0 调用相关的 SFC，然后等待 BUSY =0。只有这样才能重新释放所有连接资源。

只能在有 SFC“X SEND”、“X PUT”或“X GET”的通信端点上才可以调用 SFC69“X—ABORT”，通过 REQ =1 来激活终止的连接。

SFC69“X_ ABORT”的输入输出参数如表 7-9 所示。

表 7-9　SFC69“X_ ABORT”的输入输出参数

参数符号	参数类型	数据类型	参 数 解 说
REQ	INPUT	BOOL	=1 使能释放通信动态连接
DEST_ID	INPUT	WORD	对方(通信伙伴)的 MPI 地址
RET_VAL	OUTPUT	INT	如果在该功能执行过程中出错,则返回值包含相应的错误代码
BUSY	OUTPUT	BOOL	=1 正在执行释放指令, =0 执行释放完毕

无组态连接通信方式中的双边编程通信方式：双边编程通信方式，也就是本地与远程两边都要编写通信程序，发送方用 SFC65 来发送数据，接收方用 SFC66 来接收数据，这些系统功能只有 S7-300/400 才有，因此双边编程通信方式只能在 S7-300/400 之间进行，不能与 S7-200 通信。

下面以实例来说明 CPU 315-2DP（4 号站）与 CPU 315-2DP（3 号站）通过双边编程实现 MPI 通信。

1）本例需要的硬件和软件。

所需的硬件：CPU315-2DP（6ES7 315-2AGIO-OABO）、CPU315-2DP（6ES7 315-2AF03-OABO）。

所需的软件：STEP 7 V5.3。

2）网络配置如图 7-17 所示。

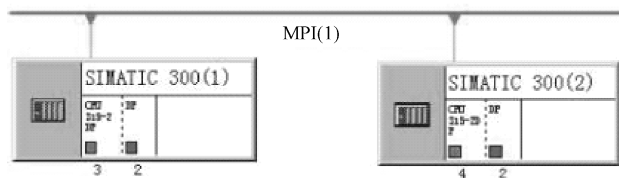


图 7-17　网络配置

3）新建一个项目“双边编程”，然后插入两个 S7-300 站及组态 MPI 网络，方法与图 7-5 ~ 图 7-7 所示相同，此处不再赘述，并在 4 号 OB5 中编写如图 7-18 所示程序，查看 S7 站点出现如图 7-19 界面。

在 S7-300（3 号站）的 OB1 中编写程序如图 7-20 所示。

在图 7-20 中，接收发送到 4 号站的数据包，数据包里通过作业标志符来分辨，而不管是哪个 CPU 发送过来的数据包。本例将接收到作业标志符为 2 的数据包。

当 M0.0 = 0 时，通过监控 M10.0 及 MD102 可以判断有没有新的数据包发送过来，当 M10.0 = 1 时，MD102 显示的就是新发送来的数据包的作业表示符。

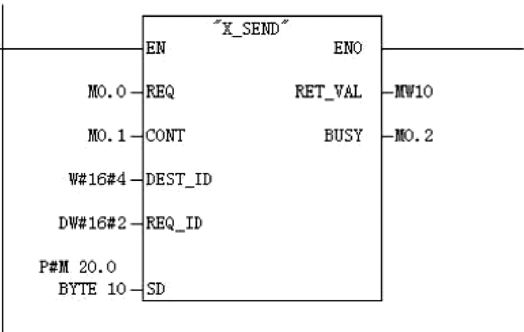
3. 单边编程通信方式

单边编程通信方式非常适合 S7-300/400 与 S7-200 CPU 间的 MPI 通信。在 S7-300/400 中使用 SFC68“X—PUT”发送数据，使用 SFC67“X GET”来接收数据。

S7-300/400 中没有 V 区，S7-200 只有 V 区而没有数据块 DB。如果对 S7-200 的 V 区进行读写，那么就要在 S7-300/400 中用 DBI 定义，也就是说 S7-200 的 V 区对应 S7-300/400 的 DBI 区。

OB35 : 每100ms调用一次

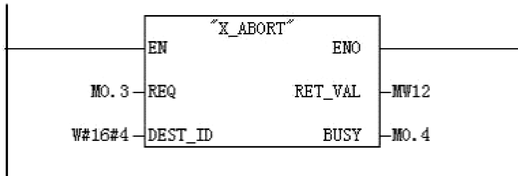
程序段?1：发送



符号信息：

X_SEND SFC65 -- Send Data to a Communication Partner outside the Local S7 Station

程序段?2：断开连接



符号信息：

X_ABORT SFC69 -- Abort an Existing Connection to a Partner outside the Local S7 Station

图 7-18 S7-300（4 号站）的 OB35 中编写程序

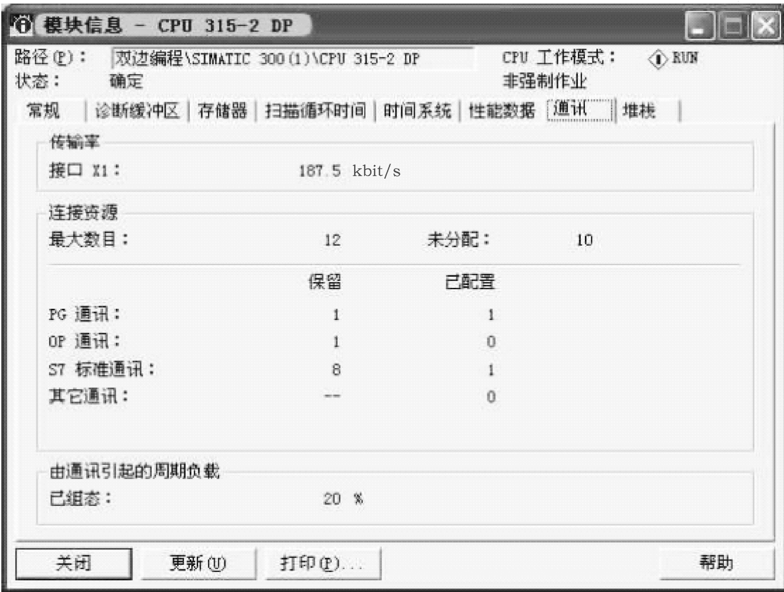
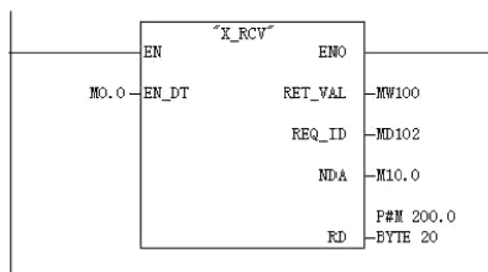


图 7-19 在线查看 S7 的连接

OB1 : "Main Program Sweep (Cycle)"

程序段?1： 标题：



符号信息：

X_RCV　SFC66　-- Receive Data from a Communication Partner outside the Local S7 Station

程序段?2： 标题：

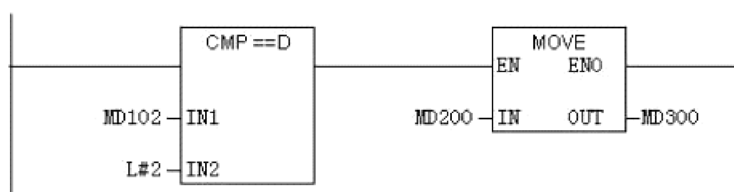


图 7-20　在 S7-300（3 号站）的 OB1 中编写程序

利用 EM277 模块也可以实现 S7-300/400 与 S7-200 之间的 MPI 通信。把 EM277 的拨码开关设为 2，代表 S7-200 在 MPI 网络上的地址是 2，与直接使用 S7-200 的编程口进行 MPI 通信实现的功能是一样的。

【例】 下面以 S7-200 与 S7-300 CPU 进行 MPI 通信，S7-300 CPU 作为客户机，S7-200 CPU 作为服务器（S7-200 CPU 只能作为服务器）为例进行说明。

1）本例需要的硬件和软件：所需的硬件：CPU315-2DP（315-2AGIO-OABO）、CPU 226（216-28022-OXBO）；

所需的软件：STEP 7 V5.3、STEP 7 MicroWIN V4.0。

2）网络配置如图 7-21 所示。

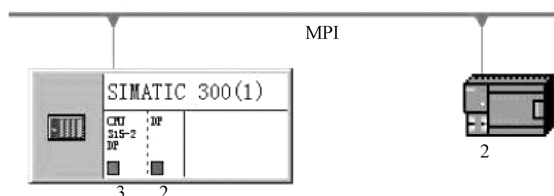


图 7-21　S7-300 与 S7-200 之间的 MPI 通信网络配置

3）设置 S7-200 的 MPI 网络。

使用 STEP 7-Micro/WIN V4.0 建立一个文件“单边编程”，选好使用的 S7-200 的型号，打开通信口，如图 7-22 所示。

设置 S7-200 CPU 端口 0 的 PLC 地址为 2、波特率为 187.5kbit/s，然后确认并下载到 CPU 中。



图 7-22 设置 S7-200 通信口参数

4）使用 STEP 7 V5.3 新建项目“单边编程”，然后插入 S7-300 站并完成 CPU 硬件组态，如图 7-23 所示。

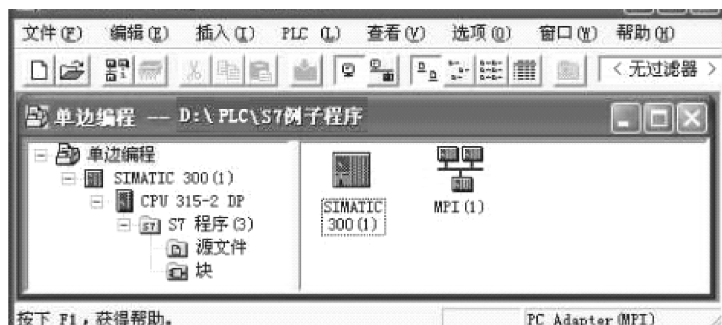


图 7-23 新建名为“单边编程”的项目

5）在硬件组态界面，如图 7-24 所示，双击机架上的 CPU，在常规界面里单击“属性”，设置 MPI 站号（此处为 3）及波特率（本例是 187.5 kbit/s）。

6）在 S7-200 的 OBI（主程序）中编写程序，如图 7-25 所示。程序的意思是把 SMB0 的状态传送到 VB0 中。

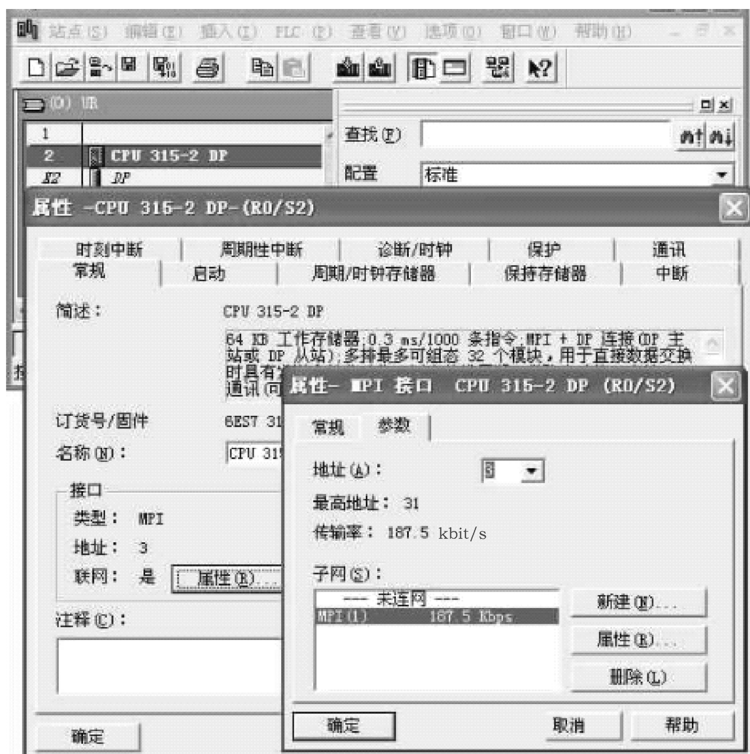


图 7-24　设置 S7-300 的 MPI 参数

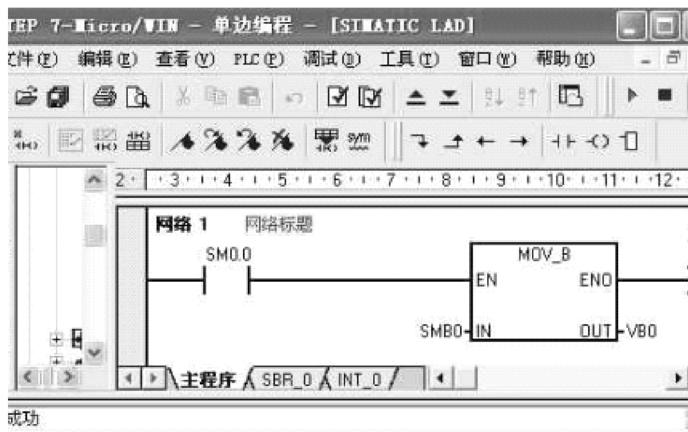


图 7-25　在 S7-200 的 OB1（主程序）中编写程序

7）在 S7-300 的 OB35 中编写程序，如图 7-26 所示。

8）当 M0.0 = 1 时，把 DB1. DBB1 的数据传送到 S7-200（2 号站）的 QB0 中。看到 S7-200 的 QB0 的状态与 SMB0 的状态几乎相同。

4. 组态连接通信方式

如果交换的信息量较大时，可以选择组态连接通信方式，这种通信方式只能在 S7-300 与 S7-400 或 S7-400 与 S7-400 之间进行。在 S7-300 与 S7-400 之间通信时 S7-300 只能做服务器，S7-400 只能做客户机；在 S7-400 与 S7-400 之间进行通信时，任意一个 CPU 都可以

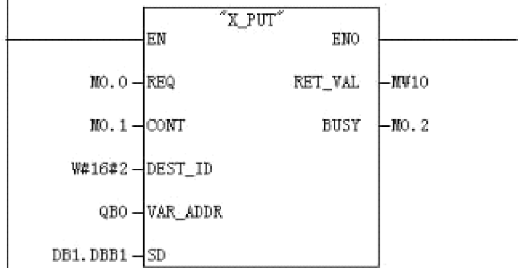
做服务器或客户机。客户机 CPU 通过调用系统功能块实现通信，数据包最大长度为 160 个字节。

S7-400 通过使用 SFB15/FB15 “PUT”，可以将数据写入到远程 CPU。输入 REQ 的上升沿处启动 SFB15/FB15。启动 SFB15/FB15 的过程中，将指向要写入数据区域（ADDR i）的指针和数据（SD i）发送到伙伴 CPU。

启动 SFB15/FB15 的过程中，远程伙伴 CPU 将所需要的数据保存在随数据一起提供的地址下面，并返回一个执行确认。如果启动 SFB15/FB15 的过程中没有产生任何错误，则在下一个 SFB/FB 调用时，通过状态参数 ONE 来指示，其数值为 1。SFB15/FB15 必须要确保通过参数 ADDR i 和 SD i 定义的区域在编号、长度和数据类型方面相互匹配。只有在最后一个作业完成之后，才能再次激活写作业。SFB15/FB15 “PUT” 输入/输出参数如表 7-10 所示。

OB35 : 每100ms调用一次

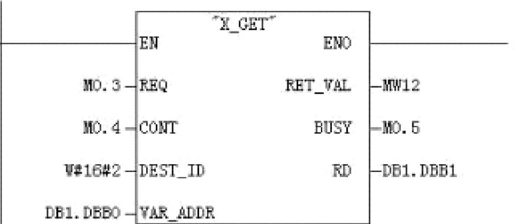
程序段?1：发送



符号信息：

X_PUT SFC68 -- Write Data to a Communication Partner outside the Local S7 Station

程序段?2：接收



符号信息：

X_GET SFC67 -- Read Data from a Communication Partner outside the Local S7 Station

图 7-26 在 S7-300 的 OB35 中编写程序

表 7-10 SFB15/FB15 “PUT” 输入/输出参数

参数符号	参数类型	数据类型	参 数 说 明
REQ	INPUT	BOOL	请求使能,在上升沿激活数据交换
ID	INPUT	WORD	寻址参数 ID,指向本地连接描述(由 STEP7 连接组态指定)
DONE	OUTPUT	BOOL	DONE 状态参数: 0:作业还未启动或仍然在运行 1:作业已经无错地执行完毕
ERROR	OUTPUT	BOOL	ERROR 和 STATUS 状态参数,出错显示 当 ERROR = 0 时,STATUS 的数值: STATUS = 0000H:既不是警告也不是出错 STATUS≠0000H:警告,STATUS 提供详细信息 当 ERROR = 1 时表示出错,STATUS 提供关于错误类型的详细信息
STATUS	OUTPUT	WORD	
ADDR_i(1≤i≤4)	IN_OUT	ANY	指针,指向伙伴 CPU 中要写入的数据区域
SD_i(1≤i≤4)	IN_OUT	ANY	指针,指向本地 CPU 中要发送的数据区域

S7-400 通过使用 SFB14/FB14 “GET” 从远程 CPU 中读取数据。输入 REQ 的上升沿处启动 SFB14/FB14。在启动 SFB14/FB14 的过程中，将要读取的区域的相关指针（ADDR i）发送到伙伴 CPU，远程伙伴返回此数据。在下一个 SFB14/FB14 调用前，已接收的数据被复制到组态的接收区（RD i）中。必须要确保通过参数 ADDR i 和 RD i 定义的区域在长度和

数据类型方面要相互匹配。通过状态参数 NDR 数值为 1 来指示此作业已完成。

只有在前一个作业已经完成之后，才能重新激活读作业。SFB14/FB14 输入/输出参数如表 7-11 所示。

表 7-11　SFB14/FB14 输入/输出参数

参数符号	参数类型	数据类型	参 数 说 明
REQ	INPUT	BOOL	请求使能，在上升沿激活数据交换
ID	INPUT	WORD	寻址参数 ID，指向本地连接描述（由 STEP7 连接组态指定）
DONE	OUTPUT	BOOL	DONE 状态参数： 0：作业还未启动或仍然在运行 1：作业已经无错地执行完毕
ERROR	OUTPUT	BOOL	ERROR 和 STATUS 状态参数，出错显示 当 ERROR = 0 时，STATUS 的数值： STATUS = 0000H：既不是警告也不是出错 STATUS≠0000H：警告，STATUS 提供详细信息 当 ERROR = 1 时表示出错，STATUS 提供关于错误类型的详细信息
STATUS	OUTPUT	WORD	
ADDR_i（1≤i≤4）	IN_OUT	ANY	指针，指向伙伴 CPU 中要读取的数据区域
SD_i（1≤i≤4）	IN_OUT	ANY	指针，指向本地 CPU 中要写入的数据区域

下面以 S7-300 与 S7-400 之间通信，S7-300 做服务器，S7-400 做客户机为例进行介绍。

【例】　组态连接实例。

1）本例所需的硬件和软件。

所需的硬件：CPU414-2DP（6ES7 414-2XJ01-OABO）、CPU315-2DP（6ES7 315—2AF03-OABO）；

所需的软件：STEP7 V5.3。

2）网络配置，配置如图 7-27 所示。

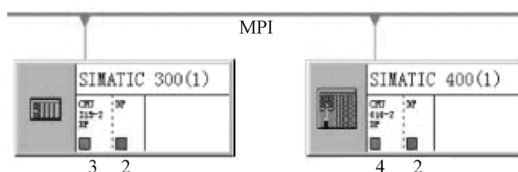


图 7-27　网络配置

3）新建一个项目“组态连接”，如图 7-28 所示。

4）然后插入两个站及组态 MPI 网络，操作方法如图 7-5 ~ 图 7-7 所示。

5）在管理界面，单击项目名称“组态连接”，双击右边的“MPI（1）”图标，将弹出 MPI 网络总线及已组态挂在网络上的站。右键单击 CPU414-2DP 的 MPI 接口，单击“插入新连接”，如图 7-29 所示。

6）在插入新连接界面中，选择如图 7-30 所示“CPU3 15-2DP”作为本项目连接对象，并选择连接类型为“S7 连接”，然后单击“应用”，将弹出如图 7-31 所示 S7 新连接属性界面，选择如图所示的参数。

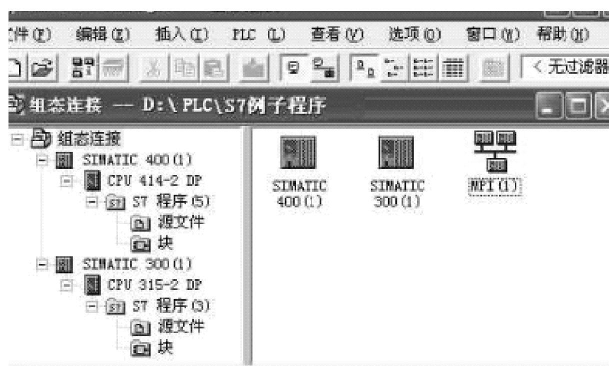


图 7-28 新建项目

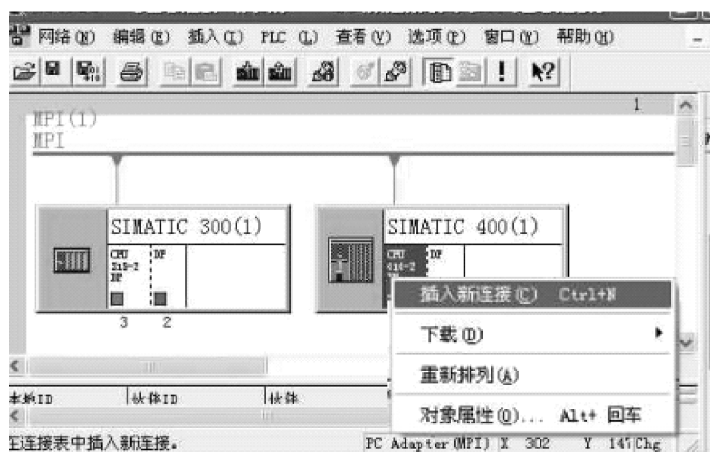


图 7-29 插入新连接



图 7-30 选择组态连接伙伴及连接方式



图 7-31　填写 S7 MPI 连接参数

7）在 MPI（1）界面里单击“ ”，编译并保存当前新建的连接参数。这样 MPI 网络组态参数全部设置完毕，如图 7-32 所示。

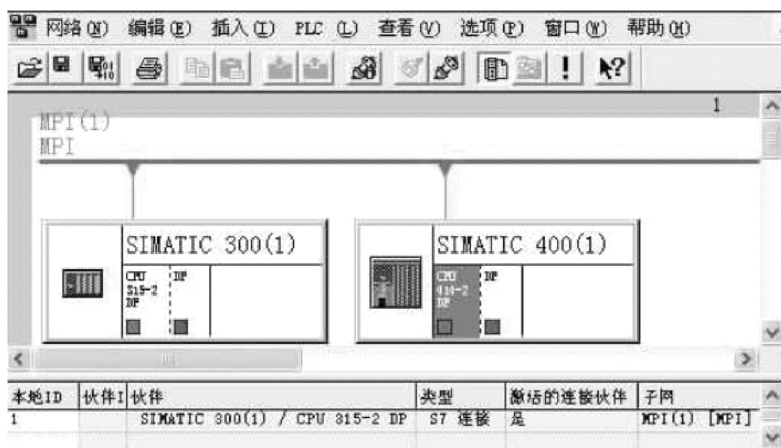


图 7-32　编译保存组态连接

8）编写程序。在 S7-400 CPU 的 OB1 里编写程序，如图 7-33 所示。

当 M0.0 上升沿时，从通信伙伴处读取数据到本地 CPU（ADDR i→RDi），即读取 S7-300（3 号站）MB10～MB19 的数据到 S7-400（4 号站）MB100～MB109 中。

当 M0.3 上升沿时，从本地 CPU 发送数据到通信伙伴处（SDi→ADDR i），即把 S7-400（4 号站）MB200～MB205 的数据发送到 S7-300（3 号站）MB20～MB25 中。

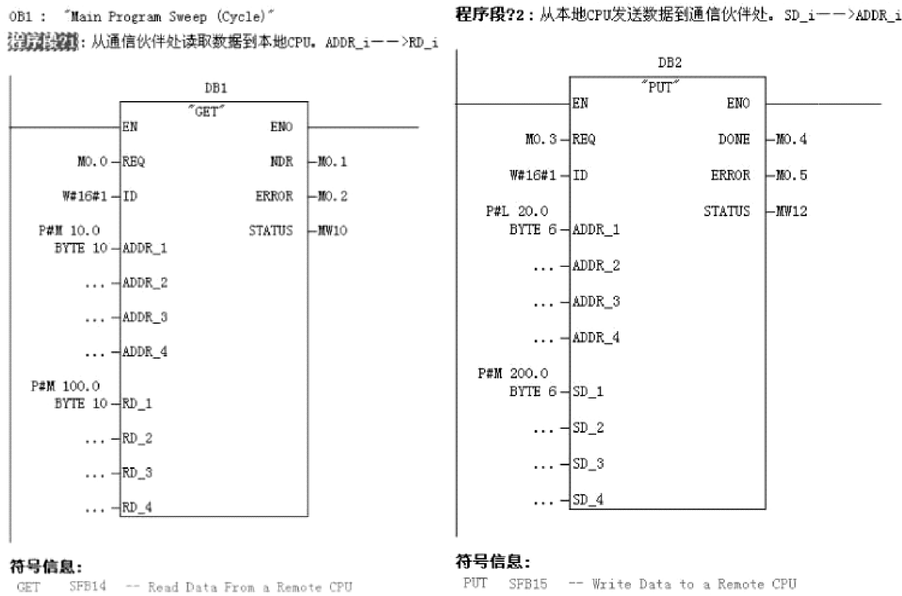


图 7-33　在 S7-400 CPU 的 OB1 里编写的程序

7.3　PROFIBUS 的结构与硬件

7.3.1　PROFIBUS 概述

工业以太网符合 IEEE 802.3（以太网）国际标准的局域和单元网络，设计用于现场级的工业环境，可以实现自动化部件之间的连接，以及自动化部件与 PC/工作站之间的连接，并可连接无线通信部件。PROFINET 是一种基于工业以太网的开放式标准，支持从公司管理层到现场层之间各设备的衔接。

工业以太网可实现范围广泛的开放式网络解决方案，是已被证实并被世界各地所接受的工业标准。

工业以太网是世界范围内局域网应用中的领先网络，其市场占有率超过 80%，并在不断增长。工业以太网具有以下优点：

1）连接简捷，调试快速。

2）灵活性高，可扩展现有的设备而不影响其运行。

3）是公司范围内实现联网的基础（纵向集成）。

4）是因特网服务的基础。

5）采用冗余网络拓扑结构，具有高可用性。

6）通过交换机技术其性能可伸缩，通信性能几乎无限制。

7）用于不同应用领域的联网，例如办公环境与生产环境。

8）利用 SCALANCE W，通过与 WAN（广域网）或 IWLAN（工业无线局域网）连接，实现公司范围内的通信。

9）利用 SCALANCE W，实现 WLAN 或 IWLAN 与移动站无故障连接。持续的兼容性开发，投资安全。

10）采用简单、有效的信令概念，持续监控网络部件。

11）在工业以太网中，可实现工厂范围内的时间同步。这样，在订购整个工厂的备件时，就可以做到有的放矢。

12）通过西门子 SCALANCE S 所实现的安全功能，能够保护网络与数据。

RJ-45 电缆及 TP Cord 连接头被广泛应用于工业以太网。TP Cord 线序如图 7-34 所示，RJ-45 电缆的两种接法如图 7-35 所示。

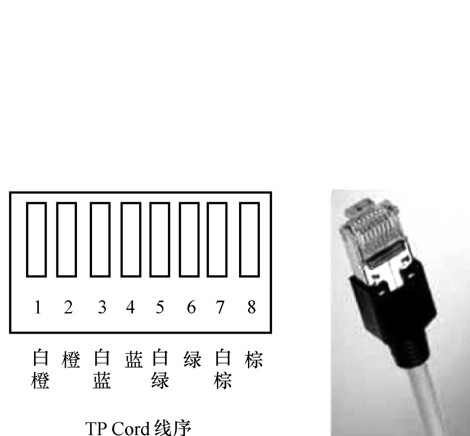


图 7-34　TP Cord 线序

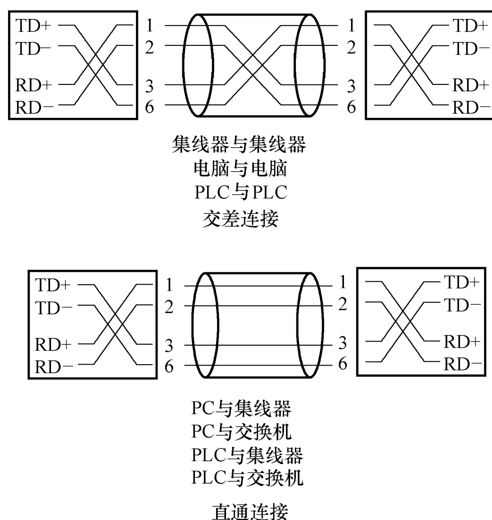


图 7-35　RJ-45 电缆的两种接法



注意：

配置 CPU31x-2PN/DP 的 PN 接口时，当 PROFINET 接口偶尔发生通信错误时，该如何处理？

请确定以太网（PROFINET）中的所有组件（转换）都支持 100Mbit/s 全双工基本操作。避免中心分配器割裂网络，因为这些设备只能工作于半双工模式。

7.3.2　基本 PROFIBUS 的项目实例

接下来我们以 S7-200 作为客户机及服务器和 S7-300/400 作为客户机及服务器的组态方法及步骤。

【例】　以太网实例 1

1. 配置的硬件及软件

所需的硬件：CPU226；CPU315-2DP；CP243-I IT　S7-200 以太网模块；CP343-I IT S7-300 以太网模块。

所需的软件：STEP7 V5.3；STEP 7 MicroWIN V4.0；SIMATIC NET V6.2。

2. 网络配置

S7-200 为服务器、S7-300 为客户机的以太网通信网络配置如图 7-36 所示。

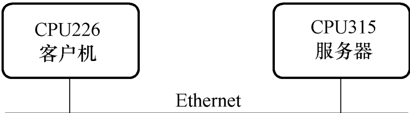


图 7-36 以太网通信网络配置

（1）S7-200 客户机：

1）首先使用 STEP 7-Micro/WIN V4.0 建立一个新项目“以太网 1”，设置 CPU 类型，如果在联机状态可以读取 CPU 类型，如图 7-37 所示。

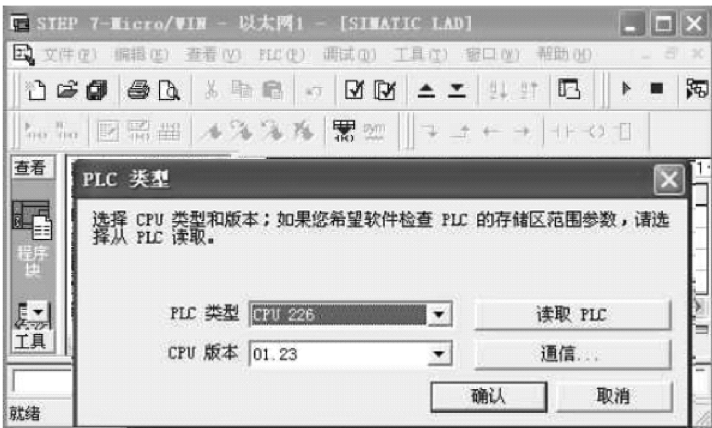


图 7-37 设置 CPU 类型

2）然后单击“工具”→“以太网向导”，如图 7-38 所示，准备进行客户机的以太网组态。



图 7-38 打开以太网向导

3）选择以太网模块的位置：直接选择，如果在联机状态可以单击“读取模块”，如图 7-39 所示。

4）设置以太网模块的地址有两种方法：

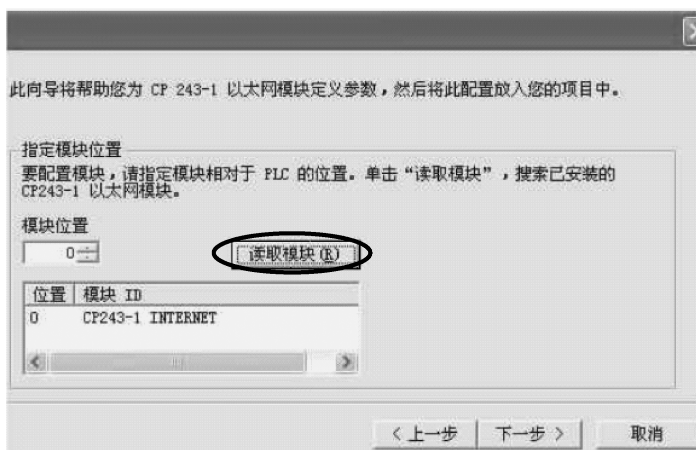


图 7-39　设置以太网模块位置

① 人工输入：在“IP 地址”中输入模块 IP 地址，或单击“ ”图标从列表中选择一个模块 IP 地址。输入了网掩码和网关地址。若本例使用静态 IP 地址，如图 7-40 所示。

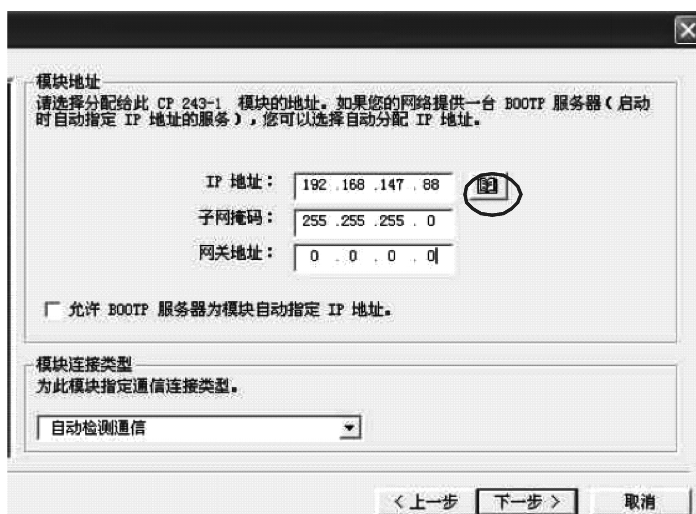


图 7-40　设置以太网模块地址等参数

② 选择“允许 BOOTP 服务器为模块自动指定 IP 地址”复选框，允许以太网模块在启动时从 BOOTP 服务器（根据 MAC 地址）获取 IP 地址、网关地址和子网掩码。如果选择该选项，则 IP 地址、子网掩码和网关地址方框无法使用（如果有 DHCP 服务器可以选择自动指定 IP）。还要为模块指定通信连接类型，可以从以下可能的通信连接类型进行选择：

自动检测通信（默认值）；

全双工 100Mbit/s 通信；

半双工 100Mbit/s 通信；

全双工 10Mbit/s 通信；

半双工 10Mbit/s 通信。



注意：

半双工（Half Duplex）

在半双工形式下数据传送是双向的，但任何时刻只能由其中的一方发送数据，另一方接收数据，即数据从 A 站发送到 B 站时，B 站只能接收数据；数据从 B 站发送到 A 站时，A 站只能接收数据，如图 7-41 所示。

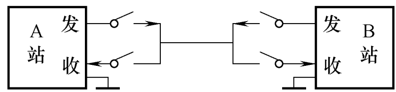


图 7-41 半双工形式

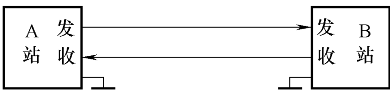


图 7-42 全双工形式

全双工（Full Duplex）

在全双工形式下数据传送也是双向的，允许双方同时进行数据双向传送，即可以同时发送和接收数据，如图 7-42 所示。

5）在图 7-43 中，输入以太网模块的输出内存地址（Q 地址）。智能模块的命令字节是指定给模块的 Q 字节（输出字节）。如果在该向导中处于连接状态，使用读取模块位置，输出内存地址会自动显示。

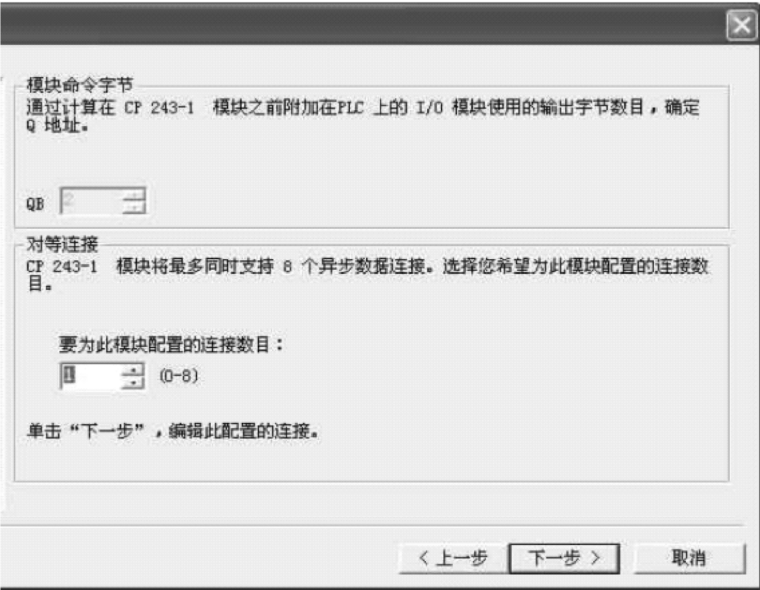


图 7-43 选择“对等连接”数

6）选择“对等连接”的数量是指定希望为以太网模块配置的连接数目，本例是 1。以太网模块最多支持 8 个异步并行连接，这 8 个对等连接是在 STEP 7-Micro/WIN 和以太网模块之间连接之外附加的连接。

7）如果选择配置某个连接，当单击“下一步”时，会显示“配置连接”界面，如图 7-44 所示，选择客户机及写上服务器的 IP 及 TSAP。

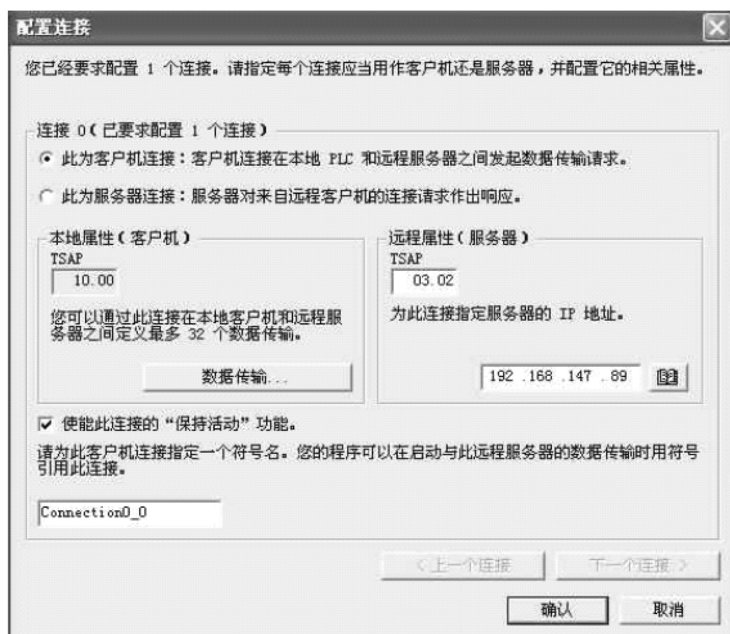


图 7-44　配置客户机的连接

在图 7-44 中，需要将每个连接定义为客户机连接或服务器连接。

8）如果选择客户机连接：客户机连接请求在本地 PLC 和远程对象之间执行数据传送。每个客户机连接可能包含 1 ~ 32 个数据传送（本例选择客户机连接）。

如果选择服务器连接：服务器连接从远程客户机接收连接请求，可将服务器配置为从任何客户机或仅限指定的客户机接受连接。



注意：

如何通过 PROFIBUSDP 用功能块实现在主、从站之间实现双向数据传送？

在主站 PLC 可以通过调用 SFC14“DPRD_ DAT“和 SFC15“DPWR_ DAT“来完成和从站的数据交换，而对于从站来说可以调用 FC1“DP_ SEND“和 FC2”DP_ RECV“完成数据的交换。

当选择配置客户机连接时，必须要配置以下几项：

① 定义远程对象的 TSAP。

当连接的远程对象是 S7-200 PLC 时，使用以下算法确定远程 TSAP：

TSAP 的第一个字节是 0x10 + 连接数目，TSAP 的第二个字节是模块位置。

当连接远程对象是 S7-300 或 S7-400 时，使用以下算法确定远程 TSAP：

TSAP 的第一个字节是 0x10 + 连接数目，TSAP 的第二个字节代表模块架和槽位的编码数值。第二个字节的第三位是模块架，最后 5 个位是编码槽号（本例 TASP = 03.02 = 16#0302）。

② 指定远程服务器的 IP 地址：方法是以手动方式输入地址（本例选择手动方式输入地

址），或单击“[icon]”从列表中选择模块地址。

③ 选择使用“保持现用”功能。该功能使模块定期向对象发出信息，使连接保持现用。

④ 为客户机连接指定一个符号名。可以使用为当前客户机连接自动定义符号名（本例选择默认符号），也可以编辑符号名。请注意，当程序请求启动与远程对象的连接时，会使用为连接定义的符号名。

⑤ 选择定义本地 PLC 和远程对象之间的数据传送。最多可为一个特定连接配置 32 个独立的数据传送。单击“数据传输”标记，并单击“新传输”，将弹出配置 CPU 至 CPU 数据传输界面，如图 7-45 所示。

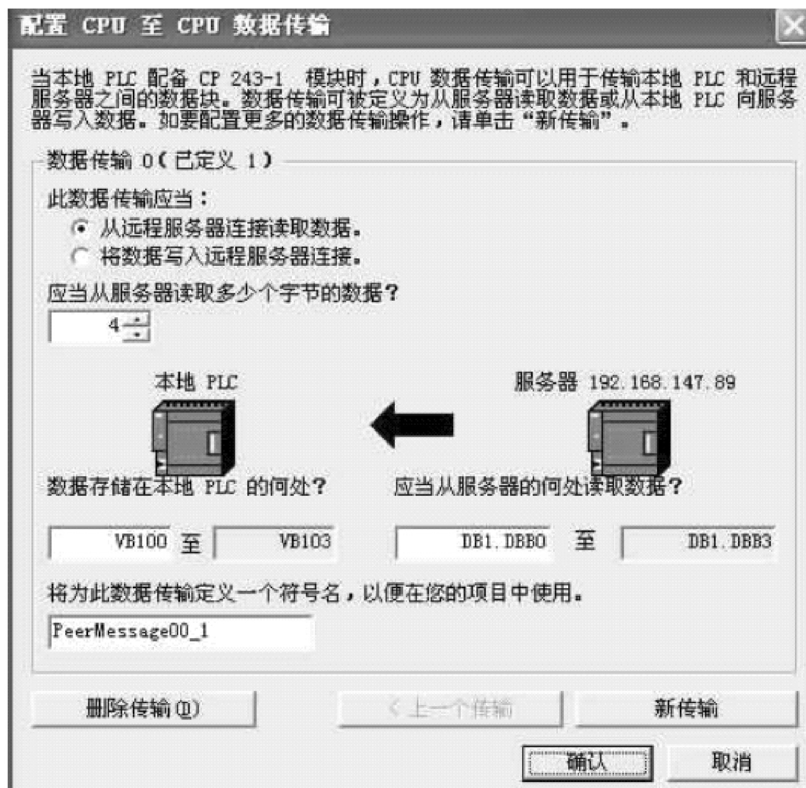


图 7-45 建立读连接

在图 7-45 中，每个 CPU 至 CPU 的数据传送可能包含一个从远程对象的数据读取或向远程对象的数据写入，必须指定希望配置的传送类型，将会自动为当前数据传送定义一个符号名。在结束向导作业时，会建立一个包含符号名的全局符号表（ETHx—SYM）。在程序请求开始与远程对象进行数据传送时需要使用这些符号名。

读取和写入操作最多可存取 212 个字节（本例是选择“读”类型，读取“4”个字节）。本地 PLC 中的地址必须是 V 内存字节地址（本例是 VB100），远程对象中的地址必须代表字节地址（本例是 DB1. DBB0）。当为 S7-300/S7-400 设备输入远程地址时，请使用 DBx. DBBy 格式。

9）单击图 7-45 中的“新传输”按钮，建立写连接，出现如图 7-46 所示画面，并定义写连接的数据地址及字节个数。


注意：

当 DP 从站不可用时，PROFIBUS 上 S7-300CPU 的监控时间是多少？

使用 CPU 的 PROFIBUS 接口上的 DP 从站操作 PROFIBUS 网络时，希望在启动期间检查期望的组态与实际的组态是否匹配。在 CPU 属性对话框中的 Startup 选项卡上给出了两个不同的时间。

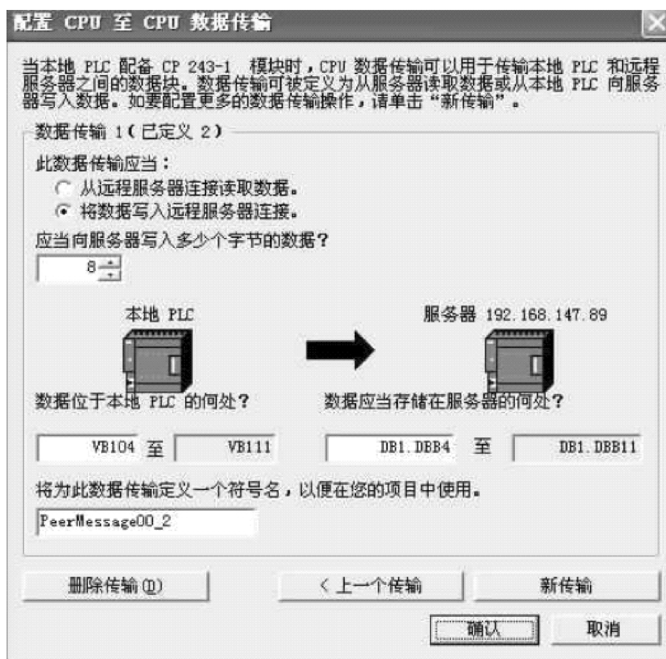


图 7-46　建立写连接

10）图 7-47 中，CRC（循环冗余检查）保护选项允许指定以太网模块检查偶然发生的配置损坏。向导为 V 内存中配置的两个数据块部分生成 CRC 值。当模块读取配置时，则重新计算该值。如果数字不匹配，配置损坏，模块不会使用该配置。

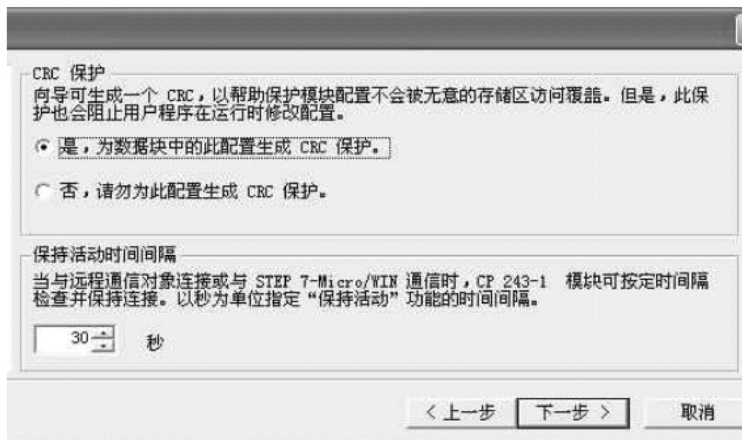


图 7-47　选择 CRC

如果选择“CRC 保护”选项，向导则不会生成“以太网重新配置”（ETHx CFG）指令，在运行时无法修改程序配置。

“保持现用”功能向对象发出一则信息，以确保连接依然现用，可以指定该时间间隔（以 s 为单位，范围是 1～32767）。

11）在图 7-48 中，以太网向导为以太网模块建立一个配置块，并将该配置存储在 PLC 的 V 内存区。该向导会自动为配置的位置指定一个起始地址，如果需要可以单击“建议地址”按钮，为配置开始地址。配置块的大小根据向导中所做的具体选择不同而异（本例选择自动配置）。

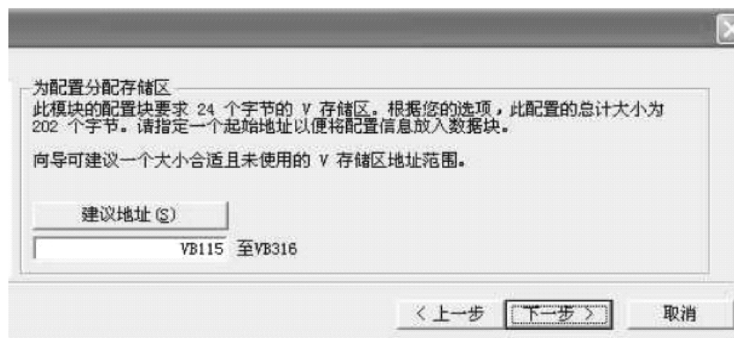


图 7-48 配置向导占用的 V 区域

12）在图 7-49 中，以太网模块向导为选择的配置（程序块和数据块）生成项目部件，并允许程序使用这些代码。但在使用前，则必须将以太网模块配置块（数据块）、系统块和程序块下载至 S7-200 CPU。

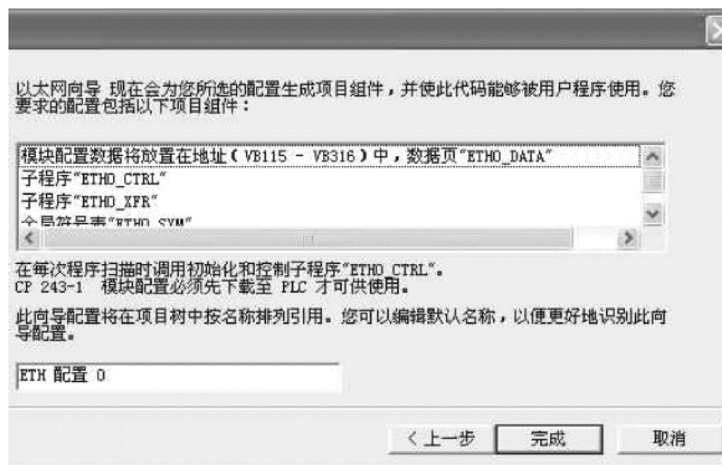


图 7-49 生成的子程序及符号

13）接着，打开左边“检视”栏的符号表，再单击“POU 符号”，出现如图 7-50 所示的画面。

符号表一栏注解里详细说明了 SBRI 和 SBR2 在编程时如何调用。单击“ETH0_ SYM”可以浏览向导生成的 ETH0_ SYM 符号，如图 7-51 所示。

符号	地址	注释
SBR_0	SBR0	子程序注释
ETH0_CTRL	SBR1	此 POU 由以太网向导生成，用于控制在位置0的 CP 243-1 模块。ETHx_CTRL（控制）指令用于使能和初始化 CP 243-1 以太网模块。此指令应在每次程序扫描时被调用，且在您的程序中仅使用一次。此模块的命令字节被指定为QB2。
ETH0_XFR	SBR2	此 POU 由以太网向导生成，用于控制在位置0的 CP 243-1 模块。ETHx_XFR（数据传输）指令用于启动 S7-200和远程连接之间的数据传输。EHTx_CTRL 指令必须在调用 ETHx_XFR 之前调用。
INT_0	INT0	中断程序注释
主程序	OB1	程序注释

图 7-50　向导生成的 POU 符号



图 7-51　向导生成的 ETH0_ SYM 符号



注意：

在编写客户机接收服务器中的数据程序，调用 SBR2 时在 Chan ID 里应该写上“VB287”在 Data 里写上“VB288”，写其他都是错的。

在编写客户机发送数据到服务器中的程序，调用 SBR2 时在 Chan ID 里应该写上“VB287”，在 Data 里写上“VB289”，写其他都是错的。

14）在组织客户机的组态中，声明建立了与服务器的发送和接收数据区的对应关系，如表 7-12 所示。

表 7-12　客户机与服务器的发送和接收数据区

客户机(S7-200)	发送/接收	服务器(S7-300)
WB100 ~ VB103	←	DB1. DBB0 ~ DB1. DBB3
WB104 ~ VB111	→	DB1. DBB4 ~ DB1. DBB111

在 S7-200 的主程序中编写实现通信功能的程序代码，内容如图 7-52 所示。

（2）服务器 S7-300 的组态

1）新建名为“以太网 1”的项目，并完成 CPU 及电源的硬件组态，如图 7-53 所示。

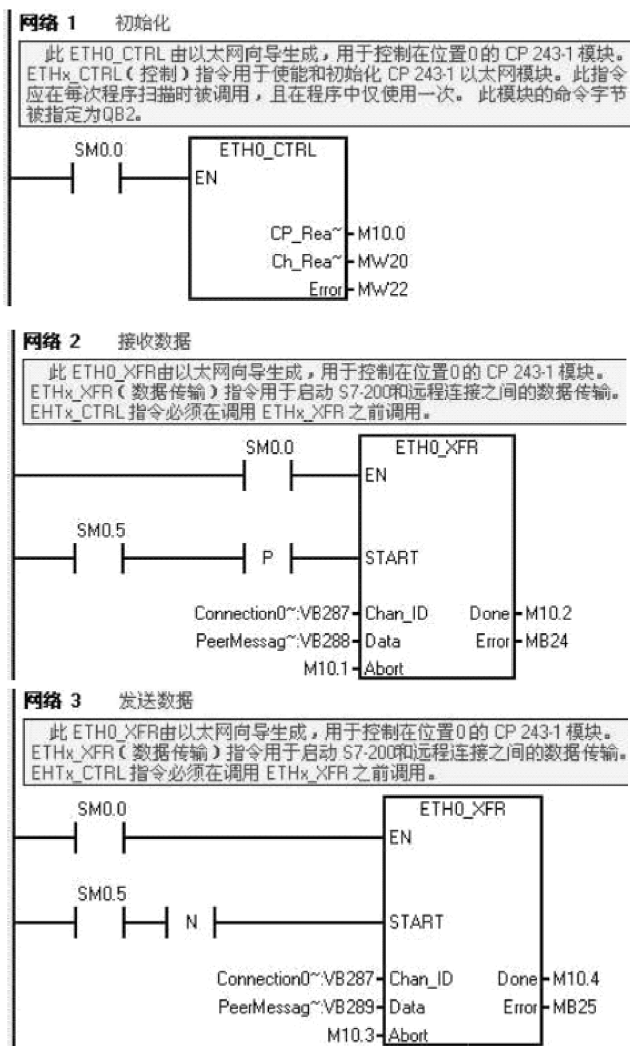


图 7-52 在 S7-200 的主程序中编写实现通信功能的程序



图 7-53 新建项目“以太网 1”

2）在硬件组态界面，在硬件目录把“CP343-1 IT”拖曳到机架的槽 4 中，如图 7-54 所示，在出现的画面（见图 7-55）中，组态 CP343-1 IT 的参数，这样就把服务器的以太网组态完毕了。

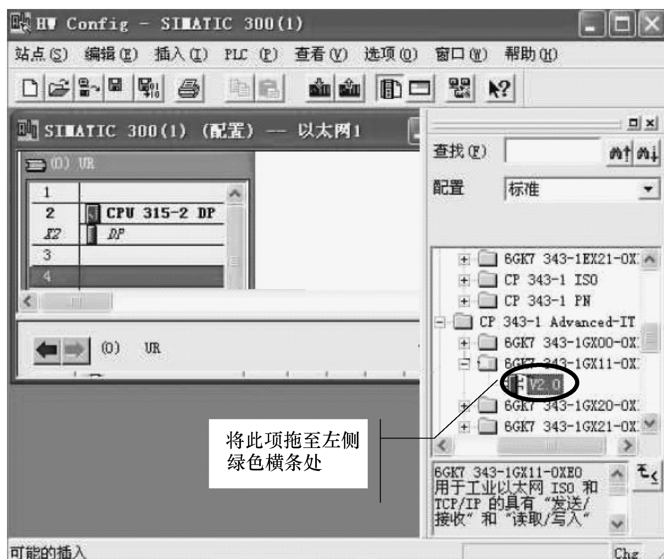


图 7-54　组态以太网模块硬件

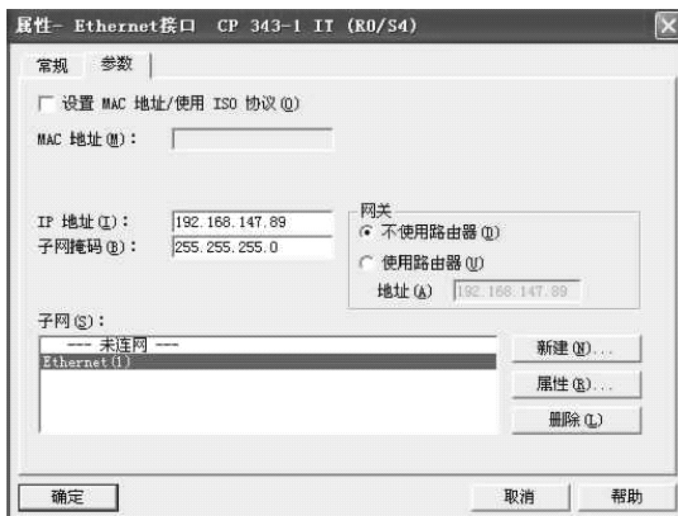


图 7-55　组态以太网模块参数

3）最后编译保存，把服务器的整个项目信息下载到 CPU 中。对于通信功能，服务器方不需要编写程序。

4）要组态以太网 CP，就必须为以太网接口分配一个 MAC 地址或 IP 地址。

每个以太网 CP 要求有一个唯一的 MAC 地址，该地址必须在组态 CP 时输入。作为惯例，制造商将此地址烧制在模块上。对于需要输入 MAC 地址的 CP，其输入框外观如图 7-56a 所示，对于配有固定出厂设置 MAC 地址的新型 CP，不需要输入 MAC 地址，其输入框外观如图 7-56b 所示。

只有在使用 ISO 协议（例如，对于 ISO 独立于网络的传输连接），或者在使用 ISO 以及 TCP/IP 时，才选中该复选框，并输入模块 MAC 地址。

在组态时若是 TCP/IP（TCP 连接、ISO-over-TCP 连接、UDP 连接）的通信类型，则不

要激活复选框。在这种情况下，不能输入 MAC 地址，而是保留烧制在模块上的地址。

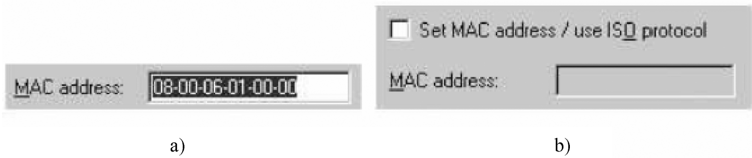


图 7-56 CP 的 MAC

只有以太网 CP 支持 TCP/IP 时，IP 参数才可见。IP 地址由 4 个十进制数字组成，每个十进制数数值范围为 0～255，十进制数字之间用点分开。

IP 地址包括：（子）网地址和伙伴地址（通常指主机或网络节点）。

子网掩码将这两个地址分开。它确定用于寻址网络和节点 IP 的 IP 地址部分。

子网掩码的设置位确定 IP 地址的网络组件。

IP 地址的前两个字节确定子网，最后的两个字节寻址节点。

网络地址是 IP 地址和子网掩码进行“与”逻辑操作的结果。

节点地址是 IP 地址和非子网掩码进行“与非”逻辑操作的结果。

IP 地址区域和所谓的“默认子网掩码”之间的关系是确定的。数字的第一个十进制数（从左到右）确定默认的（二进制）结构，即数字“1”的含义如图 7-57 所示。

IP地址(十进制)	IP地址(二进制)	地址类别	默认子网掩码
0 - 126	0xxxxxxx.xxxxxxxx....	A	255.0.0.0
128 - 191	10xxxxxx.xxxxxxxx....	B	255.255.0.0
192 - 223	110xxxxx.xxxxxxxx...	C	255.255.255.0

图 7-57 IP 地址区域和默认子网掩码间的关系

网络网关（路由器）将子网互连。发送给另一个网络的 IP 数据包必须先传送到路由器。为了启用此特性，必须为每个网络节点输入路由器地址。子网节点和网络网关（路由器）的 IP 地址可能只在子网掩码的“0”位上有所不同。

【例】 以太网通信 2。

在本例中，使用的是 S7-200 与 S7-300/400 的以太网通信，S7-200 为服务器，S7-300/400 为客户机。

（1）本例中所需的硬件及软件

所需的硬件：CPU226；CPU414-2DP；CP243-I IT　S7-200 以太网模块；CP443-I IT　S7-400 以太网模块。

所需的软件：STEP7 V5.3；STEP 7 MicroWIN V4.0；SIMATIC NET V6.2。

（2）网络配置

S7-200 为服务器，S7-400 为客户机的以太网通信网络配置如图 7-58 所示。

（3）S7-200 服务器

首先使用 STEP7 Micro/WIN V4.0 建立一个名为“以太网 2”新项目，设置 CPU 类型，如果在联机状态可以读取 CPU 类型。

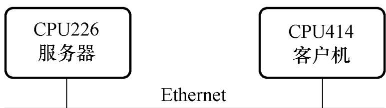


图 7-58 以太网通信网络配置

打开以太网指令向导，进行以太网通信参数设置

方法如图 7-37 ~ 图 7-43 所示。TSAP 由两个字节组成：

第一个字节定义了连接数：

1） Local TSAP（服务器）定义范围：16#02、16#10 ~ 16#FE；

2） Remote TSAP（客户机）定义范围：16#02、16#03、16#10 ~ 16#FE。

第二个字节定义了机架上 CP 槽号，本例服务器是 16#00，在 00 位置上。

客户机的 TSAP 是 10.03，如图 7-59 所示。

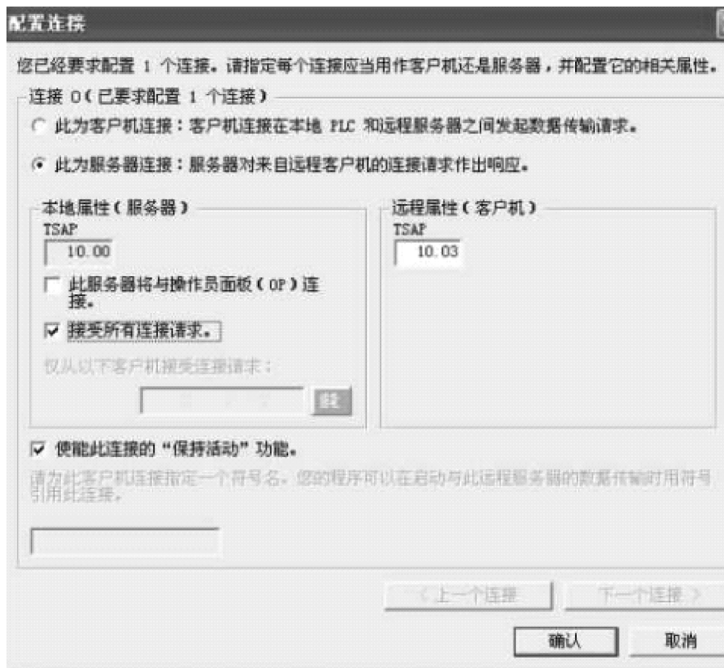


图 7-59　打开以太网向导

选择生成“CRC”，CRC 的全称是 Cycle Redundancy Check，即循环冗余码校验，如图 7-60 所示。

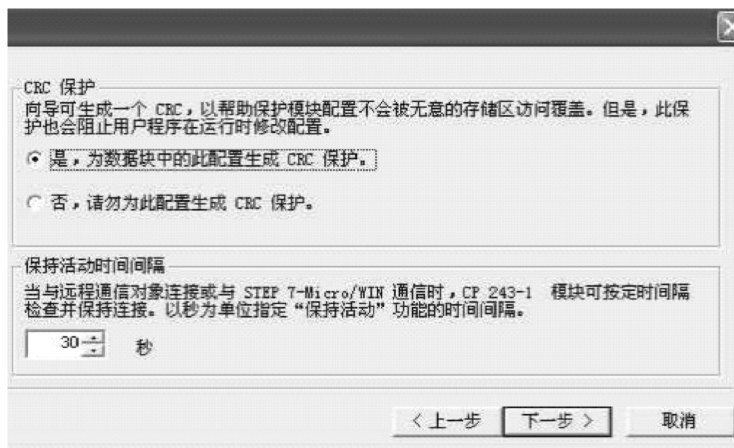


图 7-60　选择 CRC

以太网指令向导生成时自动占用系统 V 区资源，这些资源用户在编程时不能占用，请注意！如图 7-61 所示，本例占用 VBO ~ VB158。

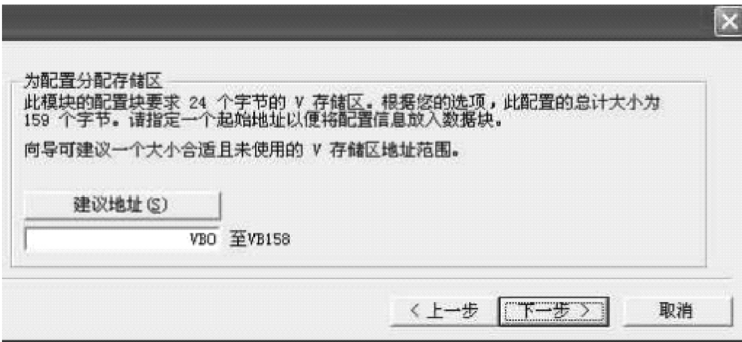


图 7-61　以太网向导占用地址区

最后可以看到以太网向导自动生成程序及占用资源信息，如图 7-62 所示。

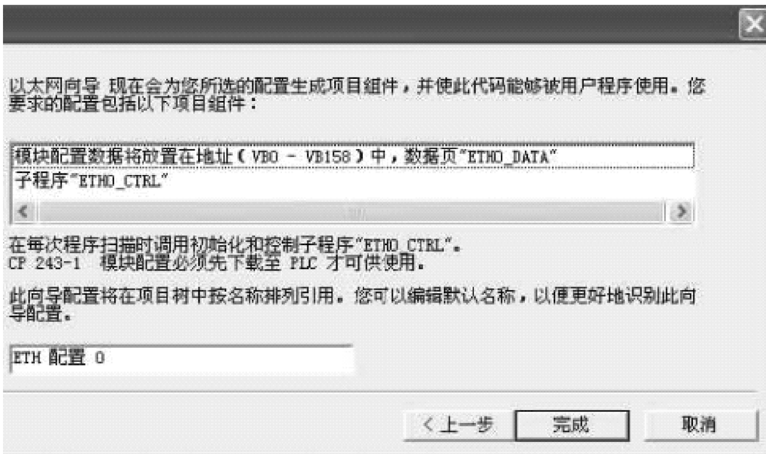


图 7-62　向导自动生成的程序

本例中在 S7-200 的 OB1 中调用向导生成的子程序如图 7-63 所示。

（4）S7-400 客户机

新建项目“以太网 2”，然后在槽架上相应位置放入 CPU 及以太网模块，如图 7-64 所示。

设置以太网模块 IP 地址、子网掩码地址。在管理界面里，单击项目名称“以太网 2”，双击右边的“Ethernet（1）”图标，打开以太网出现如图 7-65 界面。

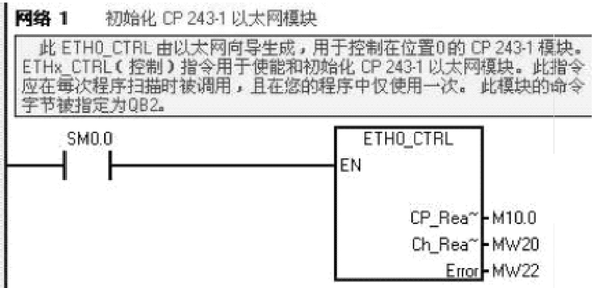


图 7-63　在 OB1 中调用子程序

在图 7-66 中，单击 S7-400 站的“CPU”字样，右键箭头标志处出现连接向导下拉表，单击“插入新连接”，出现如图 7-67 所示的新连接向导，选择“未指定”及“S7 连接”，然后单击“应用”。

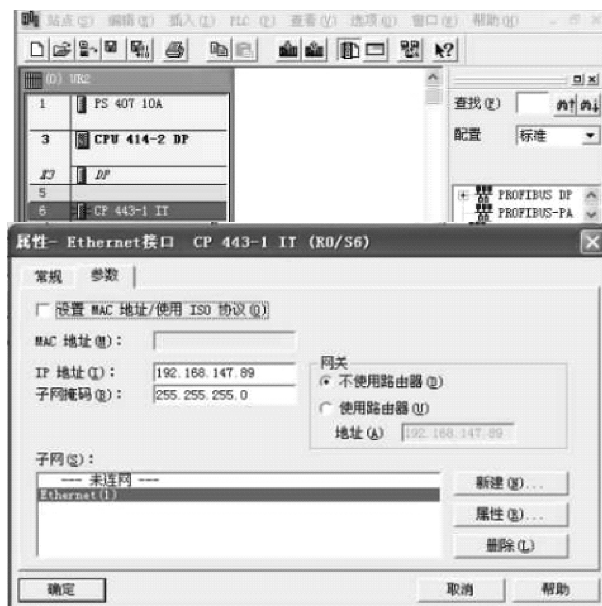


图 7-64　建立项目

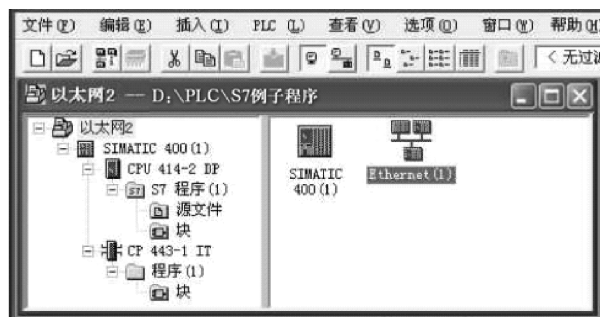


图 7-65　打开以太网



图 7-66　新增以太网



图 7-67　新连接

在图 7-68 中，在本地 ID 里写上“1”号连接，在地址里写上服务器的 IP 地址。然后单击“地址详细信息”按钮，出现如图 7-69 所示的地址详细信息设置界面，设定 TSAP。

在图 7-69 中，写上本地与伙伴的 TSAP，然后单击“确定”按钮，编译保存，客户机的以太网连接已经组态完毕。但网络上具体要发送或接收什么信息还没有表达出来，还要在客户机里编写程序把需要交换的信息编写出来，服务器不需要编写程序。

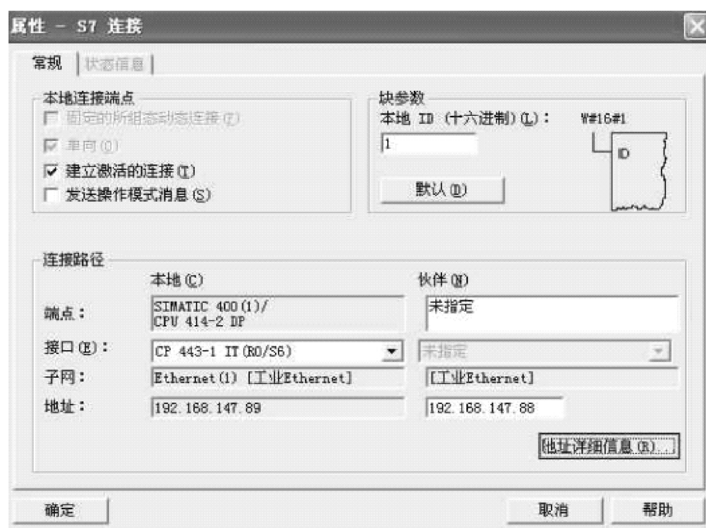


图 7-68　设定“1”号连接的参数

在客户机 OBI 里调用 SFB14，实现的功能是把服务器的数据读到客户机中；调用 SFB15，实现的功能是把客户机的数据发送到服务器。实现以太网通信功能的程序如图 7-70 所示。

在管理界面中，把整个项目的信息下载到 CPU 中，如图 7-71 所示，这样就设置完成。

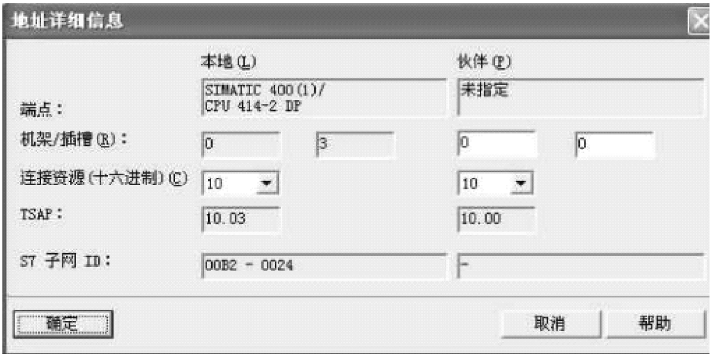
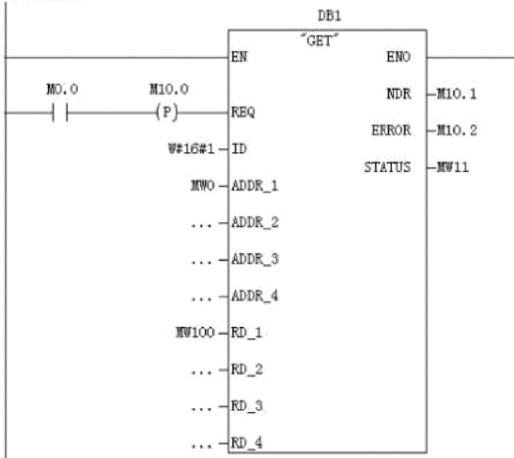


图 7-69　设定 TSAP

OB1 : "Main Program Sweep (Cycle)"S7-400客户机的通信程序

程序段?1: 每当M0.0的上升沿时，读取S7-200中MW0的数到地的MW100中。

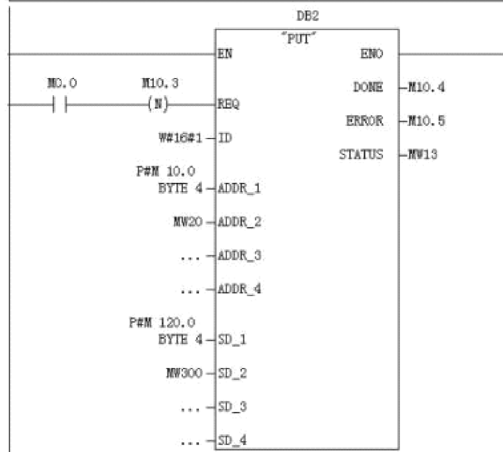


符号信息：

GET　SFB14　-- Read Data From a Remote CPU

程序段?2：把本地数据发送到远程伙伴中

每当M0.0的下降沿，将发送本地的MB120～MB124和MW300到S7200的MB10～MB14和MW20中



符号信息：

PUT　SFB15　-- Write Data to a Remote CPU

图 7-70　在客户机（S7-400）OB1 里编写的程序


SIMATIC Manager - 以太网2
文件(F) 编辑(E) 插入(I) PLC(L) 查看(V) 选项(O) 窗口(W) 帮助(H)
< 无过滤器
以太网2 — D:\PLC\S7例子程序
以太网2
SIMATIC 400(1)
CPU 414-2 DP
S7 程序(1)
源文件
块
CP 443-1 IT
程序(1)
块
系统数据
OB1
DB1
DB2
SFB14
SFB15
按下 F1，获得帮助。
PC Adapter(MPI)

图 7-71　管理界面画面

第 8 章

S7-300/400 PLC在模拟量及闭环控制系统中的应用

8.1 模拟量

8.1.1 模拟量概述

连续变化的物理量被称为模拟量，比如温度、压力、速度、流量等。我们知道 CPU 是以二进制格式来处理模拟值，其处理流程如图 8-1 所示。

模拟量输入模块的作用是将模拟过程信号转换为数字格式。

模拟量输出模块的功能是将数字输出值转换为模拟信号。

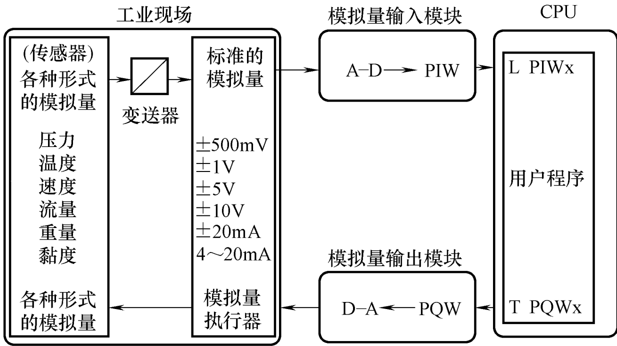


图 8-1 模拟量处理流程

模拟量输入流程是：通过传感器把物理量转变为电信号，这个电信号可能是离散性的电信号，需要通过变送器转换为标准的模拟量电压或电流信号，模拟量模块接收到标准的电信号后通过 A-D 转换，转变为与模拟量成比例的数字量信号，并存放在缓冲器里，CPU 通过“L PIWx”指令读取模拟量模块缓冲器的内容，并传送到指定的存储区中待处理。模拟量输出流程是：CPU 通过“T PQWx”指令把指定的数字量信号传送到模拟量模块的缓冲器中，模拟量模块通过 D-A 转换器，把缓冲器的内容转变为成比例的标准电压或电流信号，标准电压或电流驱动相应的执行器动作，完成模拟量控制。

前文说过，CPU 以二进制格式来处理模拟值。数字化模拟值适用于相同标称范围的输入和输出值。模拟值均为二进制补码形式的实数，符号始终设在 bit15：“0”→+，“1”→?。模拟值可能的精度如表 8-1 所示，表中以符号位对齐，未用的低位则用“0”来填补，表中的“x”

表示未用的位。

表 8-1 模拟值可能的精度

精度(位数)	分辨率		模拟值	
	十进制	十六进制	高 8 位	低 8 位
8	128	80H	符号 0000000	1xxxxxxx
9	64	40H	符号 0000000	01xxxxxx
10	32	20H	符号 0000000	001xxxxx
11	16	10H	符号 0000000	0001xxxx
12	8	8H	符号 0000000	00001xxx
13	4	4H	符号 0000000	000001xx
14	2	2H	符号 0000000	0000001x
15	1	1H	符号 0000000	00000001

8.1.2 模拟量输入模块

1. 设置模拟量输入通道的测量方法和量程

在 STEP 7 中为模拟量模块定义全部参数，找到它们后将这些参数从 STEP 7 下载到 CPU。CPU 在 STOP→RUN 切换过程中将各参数传送至相应的模拟量模块。另外，还要根据需要设置各模块的量程卡。

有两种方法设置模拟量输入通道的测量方法和量程：

1）使用量程模块和在 STEP 7 中定义模拟量模块全部参数。

2）通过模拟量模块上的接线方式，并在 STEP 7 中定义模拟量模块全部参数。

量程模块设置测量方法和量程如下：

（1）量程模块连接在模拟量输入模块旁

在安装模拟量输入模块之前，应先检查量程模块的测量方法和量程，并根据需要进行调整。模拟量模块的标签上提供了 4 种设置为“A”、“B”、“C”和“D”，如表 8-2 所示。

表 8-2 量程模块“A”、“B”、“C”和“D”的含义

量程模块设置	测量方法	测 量 范 围
A	电压	-1000 ~ 1000mV
B	电压	-10 ~ 10V
C	电流:4 线变送器	4 ~ 20mA
D	电流:2 线变送器	4 ~ 20mA

购买模拟量模块时，可以根据需要选择与模块配套的量程模块。在使用之前重新定位量程模块，设置量程模块的方法如下：

1）用螺钉旋具将量程模块从模拟量输入模块中撬出，如图 8-2 所示。将量程模块插入模拟量输入模块的插槽中（此处选择（1）号插槽），所选量程指向模块上的标记（2）。

（2）接线

模拟量输入模块支持各种传感器，如电压/电流以及电阻传感器，具体取决于硬件接线

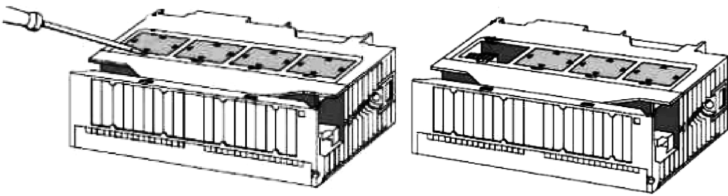


图 8-2　将量程模块放入撬开的插槽

及参数设置或量程模块安装有关。

模拟量信号电缆：使用屏蔽双绞线电缆连接模拟量信号，这样会减少干扰。线头两端的任何电位差都可能导致在屏蔽层产生等电位电流，从而产生干扰模拟信号。为防止发生这种情况，应只将电缆一端的屏蔽层接地。

1）电压传感器的连接，如图 8-3 所示。

图中的符号代表的含义如下：

M +：测量线路（正极）；

M −：测量线路（负极）；

M_{ANA}：模拟量测量电路的参考电位；

M：接地；

L +：DC24V 电源。

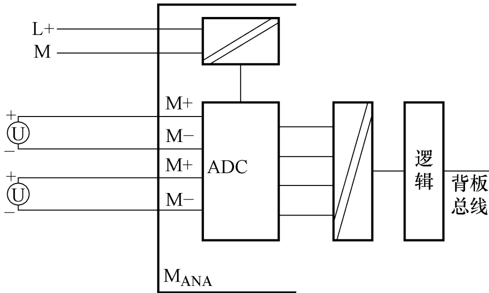


图 8-3　连接电压传感器与电气隔离模拟量输入

2）连接电流传感器。电流传感器分 2 线式和 4 线式两种。

2 线式传感器接线方法如图 8-4 和图 8-5 所示。

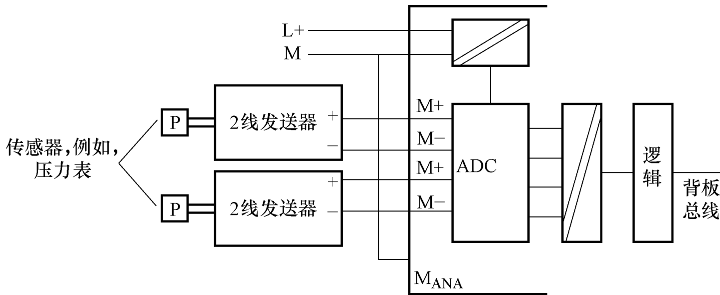


图 8-4　2 线式传感器接线

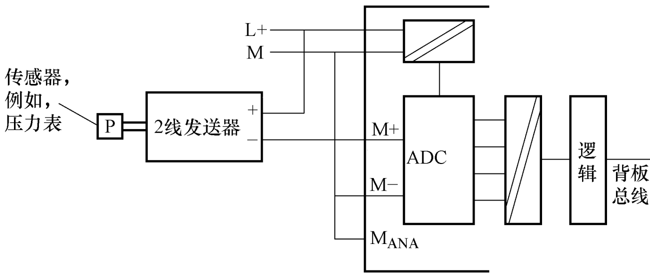


图 8-5　从 L +供电的 2 线式传感器连接到电气隔离的模拟量输入上

4 线式传感器接线方法如图 8-6 所示。

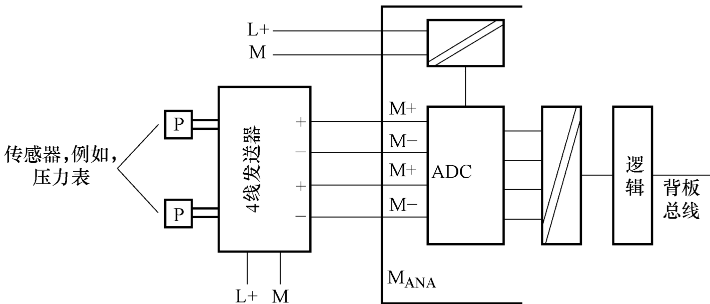


图 8-6　4 线式传感器连接到电气隔离的模拟量输入上

图 8-4 ~ 图 8-6 中的符号代表的含义如下：

M +：测量线路（正极）；　　M：接地；

M −：测量线路（负极）；　　L +：DC24V 电源。

M_{ANA}：模拟量测量电路的参考电位；

2. 输入量程的模拟值代表含义

1）范围在 ±1 ~ ±10V 电压量程内模拟值的表示如表 8-3 所示。

表 8-3　范围在 ±1 ~ ±10V 电压量程内模拟值的表示

电　压　量　程				模拟值		
±10V	±5V	±2.5V	±1V	十进制	十六进制	
11.85V	5.92V	2.963	1.185V	32767	7FFF	上溢
				32512	7F00	
11.759V	5.879V	2.940V	1.176V	32511	7EFF	上溢警告
				27649	6C01	
10V	5V	2.5V	1V	27648	6C00	正常
7.5V	3.75V	1.875V	0.75V	20736	5100	
361.7μV	180.8μV	90.4μV	34.17μV	1	1	
0V	0V	0V	0V	0	0	
				−1	FFFF	
−7.5V	−3.75V	−1.875V	0.75V	−20736	AF00	
−10V	−5V	−2.5V	−1V	−27648	9400	
				−27649	93FF	下溢警告
−11.759V	−5.879V	−2.940V	−1.176V	−32512	8100	
				−32513	80FF	下溢
11.851V	−5.926V	−2.963V	−1.185V	−32769	8000	

2）范围在 ±80 ~ ±500mV 电压量程内模拟值的表示如表 8-4 所示。

表 8-4　范围在 ±80 ~ ±500mV 电压量程内模拟值的表示

电　压　量　程			模　拟　值		
±500mV	±250mV	±80mV	十进制	十六进制	
592.6mV	296.3mV	94.8mV	32767	7FFF	上溢
			32512	7F00	
587.9mV	294.0mV	94.1mV	32511	7EFF	上溢警告
			27649	6C01	
500mV	250mV	80mV	27648	6C00	正常
375mV	187.5mV	60mV	20736	5100	
18.08μV	9.04μV	2.89μV	1	1	

（续）

电　压　量　程			模　拟　值		
±500mV	±250mV	±80mV	十进制	十六进制	
0mV	0mV	0mV	0	0	正常
			-1	FFFF	
-375mV	-187.5mV	-60mV	-20736	AF00	
-500mV	-250mV	-80mV	-27648	9400	
			-27649	93FF	下溢警告
-587.9mV	-294.0mV	-94.1mV	-32512	8100	
			-32513	80FF	下溢
-592.6mV	-296.3mV	-94.8mV	-32768	8000	

3）范围在 1 ~5V 以及 0 ~10V 电压量程内模拟值的表示如表 8-5 所示。

表 8-5　范围在 1 ~5V 以及 0 ~10V 电压量程内模拟值的表示

电　压　量　程		模　拟　值		
1 ~5V	0 ~10V	十进制	十六进制	
5.741V	11.852V	32767	7FFF	上溢
		32512	7F00	
5.704V	11.759V	32511	7EFF	上溢警告
		27649	6C01	
5V	10V	27648	6C00	正常
4V	7.5V	20736	5100	
1V +144.7μV	0V +361.7μV	1	1	
1V	0V	0	0	
	不支持负值	-1	FFFF	下溢警告
0.296V		-4864	ED00	
		-4865	ECFF	下溢
		-32768	8000	

4）范围在 ±3.2 ~ ±20mA 电流量程内模拟值的表示如表 8-6 所示。

表 8-6　范围在 ±3.2 ~ ±20mA 电流量程内模拟值的表示

电　流　量　程			模　拟　值		
±20mA	±10mA	±3.2mA	十进制	十六进制	
23.70mA	11.85mA	3.79mA	32767	7FFF	上溢
			32512	7F00	
23.52mA	11.76mA	3.76mA	32511	7EFF	上溢警告
			27649	6C01	
20mA	10mA	3.2mA	27648	6C00	正常
15mA	7.5mA	2.4mA	20736	5100	
723.4nA	361.7nA	115.7nA	1	1	
0mA	0mA	0mA	0	0	
			-1	FFFF	
-15mA	-7.5mA	-2.4mA	-20736	AF00	
-20mA	-10mA	-3.2mA	-27648	9400	
			-27649	93FF	下溢警告
-23.52mA	-11.76mA	-3.76mA	-32512	8100	
			-32513	80FF	下溢
-23.70mA	-11.85mA	-3.79mA	-32768	8000	

5）范围在 0 ~20mA 以及 4 ~20mA 电流量程内模拟值的表示如表 8-7 所示。

表 8-7 范围在 0 ~20mA 以及 4 ~20mA 电流量程内模拟值的表示

电 流 量 程		模 拟 值		
0 ~20mA	4 ~20mA	十进制	十六进制	
23.70mA	22.96mA	32767	7FFF	上溢
		32512	7F00	
23.52mA	22.81mA	32511	7EFF	上溢警告
		27649	6C01	
20mA	20mA	27648	6C00	正常
15mA	16mA	20736	5100	
723.4nA	4mA +578.7nA	1	1	
0mA	4mA	0	0	
		-1	FFFF	下溢警告
-3.52mA	1.185mA	-4864	ED00	
		-4865	EDFF	下溢
		-32768	8000	


注意：

用 S7-300 模拟量输入模块测量温度（华氏）时，可以使用模块说明文档中列出的绝对误差极限吗？

不可以直接使用指定的误差极限。基本误差和操作误差都以绝对温度和摄氏温度说明。必须乘以系数 1.8 将其转换为华氏温度单位。

例：S7-300AI8xRTD：指定的温度输入操作误差是 +/-1.0℃。当以华氏温度测量时，可接受的最大误差是 +/-1.8℉。

8.1.3 模拟量输出模块

1. 模拟量输出模块设置

模拟量输出模块的参数可以在 STEP 7 中设置。


注意：

如果未在 STEP 7 中设置任何参数，系统将使用默认参数。

模拟量输出模块的参数有诊断中断、组诊断、输出类型选择（电压、电流或禁用）、输出范围选择及对 CPU STOP 模式的响应，还可以为负载和执行器提供电源。

模拟量输出模块使用屏蔽双绞线电缆连接模拟量信号至执行器。敷设 QV 和 S+以及 M 和 S-两对信号双绞线，以减少干扰，线头两端的任何电位差都可能导致在屏蔽层产生等电位电流，进而干扰模拟信号。与输入模块一样，应只将电缆一端的屏蔽层接地。

（1）电压输出

如果使用的是电气隔离模拟量输出模块，测量电路 M_{ANA}的参考点和 CPU 的 M 端子应没有电气互连。如果测量电路 M_{ANA}的参考点和 CPU 的 M 端子间可能产生电位差 V_{iso}，务必请使用电气隔离模拟量输入模块。用等电位连接导线连接 M_{ANA}端子和 CPU 的 M 端子，以

防 V_{iso}超出限值。

如果使用的是非隔离模拟量输出模块时，务必请将测量电路的参考点 M_{ANA}与 CPU 的端子 M 互连。将 M_{ANA}端子连接到 CPU 的 M 端子。M_{ANA}和 CPU 的 M 端子间的任何电位差都可能干扰模拟信号。

模拟量模块电压输出将负载的连接，通常可以选择用 2 线或 4 线方法接线。

非隔离模拟量模块电压输出的 2 线连接到负载如图 8-7 所示。

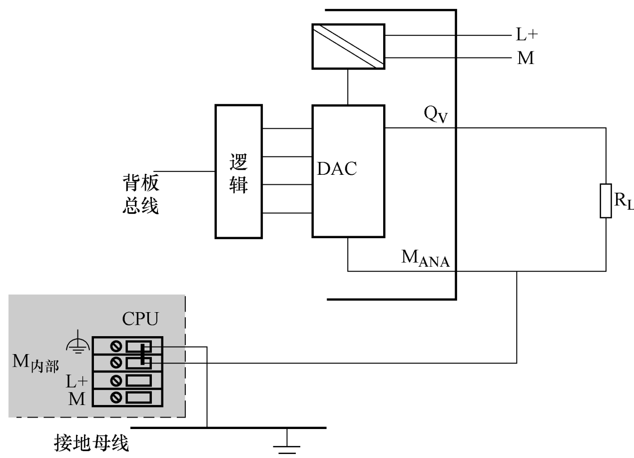


图 8-7　非隔离模拟量模块电压输出的 2 线连接到负载

使用 2 线连接时，无须连接 S + 和 S – ，但这样达不到 4 线连接的精度。将负载连接到 Q_V 端子和测量电路 M_{ANA}的参考点。

隔离模拟量模块电压输出的 4 线连接到负载如图 8-8 所示。

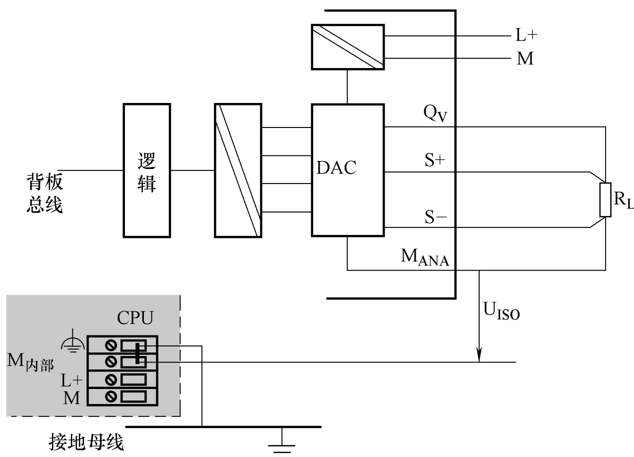


图 8-8　隔离模拟量模块电压输出的 4 线连接到负载

如果使用了 4 线方法将 S – 和 S + 检测线路直接连接到负载，获得高精度的负载，这样即可直接测量和修正负载电压。干扰和电压降可能导致在检测线路 S – 和模拟电路 M_{ANA}的参考电路间产生电位差，此电位差可能不会超出限制值。但是，也必然会对模拟信号的精度

造成负面影响。

图 8-6 和图 8-7 中符号代表的含义如下：

Q_V：模拟量输出电压；

S +：检测线路（正极）；

S −：检测线路（负极）；

M_{ANA}：模拟量电路的参考电位；

R_L：负载阻抗；

L +：DC24V 电源；

M：接地；

U_{iso}：M_{ANA}和 CPU 的 M 端子之间的电位差。

（2）电流输出

电气隔离模拟量输出模块的电流输出连接到负载如图 8-9 所示；非电气隔离模拟量输出模块的电流输出连接到负载如图 8-10 所示。

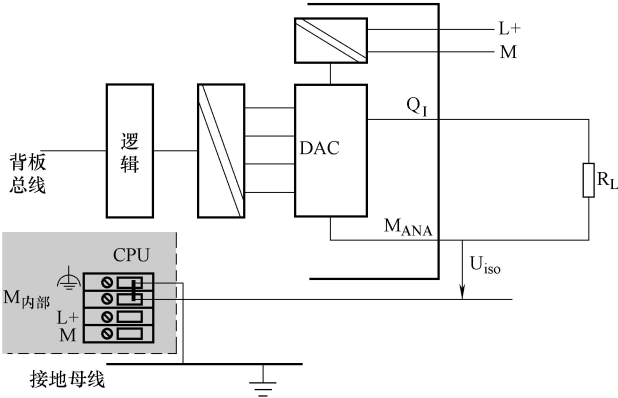


图 8-9　电气隔离模拟量输出模块的电流输出连接到负载

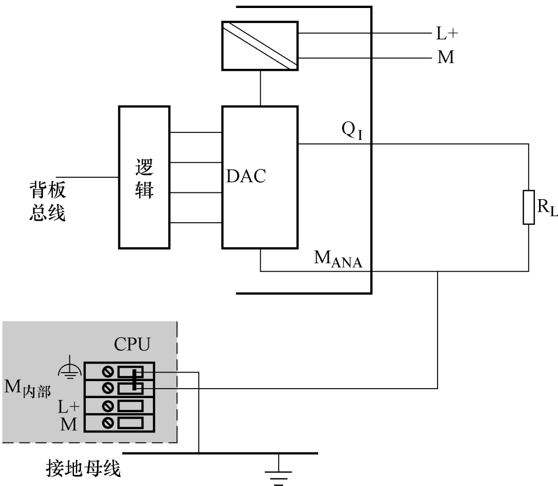


图 8-10　非电气隔离模拟量输出模块的电流输出连接到负载



提示：

请务必将负载连接到 Q_I 和电流输出的模拟量电路 M_{ANA} 的参考点。

图 8-9 和图 8-10 中符号代表的含义如下：

Q_I：模拟量输出电流；

M_{ANA}：模拟量电路的参考电位；

R_L：负载阻抗；

L+：DC24V 电源；

M：接地；

U_{iso}：M_{ANA} 和 CPU 的 M 端子之间的电位差。

2. 输出量程的模拟值代表含义

1）范围在 ±10 ~ 10V 输出范围内的模拟值表示如表 8-8 所示。

表 8-8　范围在 ±10 ~ 10V 输出范围内的模拟值表示

数　字　量			输出电压范围	
百分比	十进制	十六进制	-10 ~ 10V	
118.5149%	32767	7FFF	0.00V	上溢
	32512	7F00		
117.589%	32511	7EFF	11.76V	上溢警告
	27649	6C01		
100%	27648	6C00	10V	正常
75%	20736	5100	7.5V	
0.003617%	1	1	361.7μV	
0%	0	0	0V	
	-1	FFFF	-361.7μV	
-75%	-20736	AF00	-7.5V	
-100%	-27648	9400	-10V	
	-27649	93FF		下溢警告
-117.593%	-32512	8100	-1176V	
	-32513	80FF		下溢
-118.519%	-32768	8000	0.00V	

2）范围在 0 ~ 10V 以及 1 ~ 5V 输出范围内模拟值的表示如表 8-9 所示。

表 8-9　范围在 0 ~ 10V 以及 1 ~ 5V 输出范围内模拟值的表示

数　字　量			输出电压范围		
百分比	十进制	十六进制	0 ~ 10V	1 ~ 5V	
118.5149%	32767	7FFF	0.00V	0.00V	上溢
	32512	7F00			
117.589%	32511	7EFF	11.76V	5.70V	上溢警告
	27649	6C01			
100%	27648	6C00	10V	5V	正常
75%	20736	5100	7.5V	3.75V	
0.003617%	1	1	316.7μV	1V + 144.7μV	
0%	0	0	0V	0V	
	-1	FFFF			下溢警告
-25%	-6912	E500		0V	

（续）

数　字　量			输出电压范围		
百分比	十进制	十六进制	0～10V	1～5V	
	-6913	E4FF			下溢，输出值限制在 0V 或空闲状态
-117.593%	-32512	8100			
	-32513	80FF			
-118.519%	-32768	8000	0.00V	0.00V	

3）范围在 ±20～20mA 输出范围内的模拟值表示如表 8-10 所示。

表 8-10　范围在 ±20～20mA 输出范围内的模拟值表示

数　字　量			输出电流范围	
百分比	十进制	十六进制	-20～20mA	
118.5149%	32767	7FFF	0.00mA	上溢，空闲状态
	32512	7F00		
117.589%	32511	7EFF	23.52mA	上溢警告
	27649	6C01		
100%	27648	6C00	20mA	正常
75%	20736	5100	15mA	
0.003617%	1	1	723.4nA	
0%	0	0	0mA	
	-1	FFFF	-723.4nA	
-75%	-20736	AF00	-15mA	
-100%	-27648	9400	-20mA	
	-27649	93FF		下溢警告
-117.593%	-32512	8100	-23.52mA	
	-32513	80FF		下溢，空闲状态
-118.519%	-32768	8000	0.00mA	

4）范围在 0～20mA 以及 4～20mA 输出范围内模拟值的表示如表 8-11 所示。

表 8-11　范围在 0～20mA 以及 4～20mA 输出范围内模拟值的表示

数　字　量			输出电流范围		
百分比	十进制	十六进制	0～20mA	4～20mA	
118.5149%	32767	7FFF	0.00mA	0.00mA	上溢
	32512	7F00			
117.589%	32511	7EFF	23.52mA	22.81mA	上溢警告
	27649	6C01			
100%	27648	6C00	20mA	20mA	正常
75%	20736	5100	15mA	15mA	
0.003617%	1	1	723.4nA	4mA+578.7nA	
0%	0	0	0mA	4mA	
	-1	FFFF			下溢警告
-25%	-6912	E500		0mA	
	-6913	E4FF			下溢，输出值限制在 0mA 或空闲状态
-117.593%	-32512	8100			
	-32513	80FF			
-118.519%	-32768	8000	0.00mA	0.00mA	

8.2　闭环控制

8.2.1　闭环控制概述

西门子 S7 具有强大的闭环控制功能，在标准指令库里提供了 5 个功能块可以用于闭环控制，它们分别是：

SFB41/FB41“CONT_C”连续控制器；SFB42/FB42“CONT_S”步进控制器；SFB43/FB 43“PULSEGEN”脉冲发生器；FB58 温度连续控制器及 FB58 温度步进控制器。

单击“开始”→“Simatic”→“Step7”→“PID 控制功能赋值”，使用参数分配工具可以轻松完成 PID 控制系统的参数分配及调试工作。

8.2.2　闭环控制功能块

1. SFB41/FB41“CONT_ C”连续控制器

应用于可以使用该控制器作为 PID 固定设定值控制器，或在多循环控制中作为层叠、混料或比率控制器。

SFB41/FB41“CONT_ C”连续控制器的功能基于使用模拟信号的采样控制器的 PID 控制算法，必要时可以通过加入脉冲发生器进行扩展，为使用成比例执行机构的 2 个或 3 个步骤控制器生成脉冲持续时间调制输出信号。SFB41/FB41“CONT_ C”连续控制器的控制过程框图如图 8-11 所示。SFB41/FB41“CONT_ C”连续控制器的输入和输出参数如表 8-12 所示。

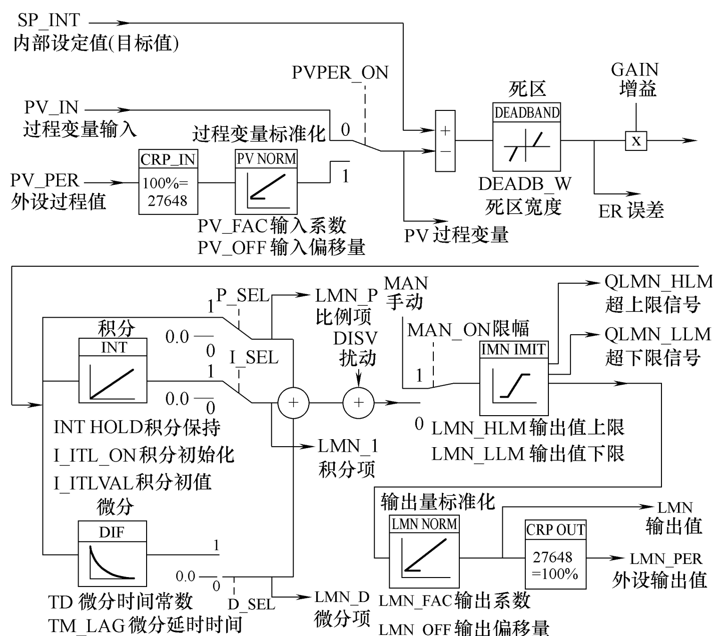


图 8-11　SFB41/FB41“CONT_ C”连续控制器的控制过程框图



注意：

应该在循环中断 OB（OB30~OB38）中定期调用这些控制器功能块，并在参数 CYCLE 中预先定义采样时间。只有定期调用块，才能正确计算控制器功能块的数值。



注意：

如何把一个 PT100 温度传感器连接到模拟输入模块 SM331？

PT100 热电阻随温度的不同其电阻值随之变化。如果有一恒定电流流经该热电阻，该热电阻上电压的下降随温度而变化。恒定电流加在接点 Ic+和 Ic-上。模拟模块 SM331 在 M+和 M-点测定电流的变化。通过测定电压就可以确定出温度。

PT100 到模拟输入组有 3 类连接：4 线连接可得到最精确的测定值。

表 8-12　SFB41/FB41“CONT_C”连续控制器的输入和输出参数

类型	参数名称	数据类型	默认值	说　明
输入参数	COM_RST	BOOL	FALSE	完全重新启动，当等于 1 时进行初始化
	CYCLE	TIME	T#1s	调用本功能块的间隔时间(采样时间)，取值范围≥1ms
	SP_INT	REAL	0.0	PID 控制目标(设定值)，取值范围在-100%～+100%
	PV_IN	REAL	0.0	采样值(小数格式)，范围在-100%～+100%
	PVPER_ON	BOOL	FALSE	采样值选择，=0 选择 PV_IN；=1 选择 PV_PER
	PV_PER	WORD	0	采样值(标准格式)，范围在-27648～+27648
	PV_FAC	REAL	1.0	PV_PER 采样值输入系数
	PV_OFF	REAL	0.0	PV_PER 采样值的偏移量
	DEADB_W	REAL	0.0	死区带宽，范围≥0.0
	GAIN	REAL	2.0	比例增益，决定控制环路的增益
	TI	TIME	T#20s	积分项时间常数，决定积分器的响应时间，范围≥CYCLE
	TD	TIME	T#10s	微分项时间常数，决定微分器的响应时间
	TM_LAG	TIME	T#2s	微分起作用的延时时间
	P_SEL	BOOL	TURE	=1 时本次环路包括比例项
	I_SEL	BOOL	TURE	=1 时本次环路包括积分项
	D_SEL	BOOL	FALSE	=1 时本次环路包括微分项
	I_ITLVAL	REAL	0.0	积分项的初始值
	I_ITL_ON	BOOL	FALSE	=1 时进行积分项初始化，选择 I_ITLVAL 为积分项初值
	INT_HOLD	BOOL	FALSE	=1 时“积分作用暂停”，“冻结”积分项的输出
	DISV	REAL	0.0	扰动输入量
	MAN_ON	BOOL	TURE	=1 时选择手动量作为环路的过程值
	MAN	REAL	0.0	MAN_ON=1 时，选择手动量输入
	LMN_HLM	REAL	100.0	环路输出上限值
	LMN_LLM	REAL	0.0	环路输出下限值
	LMN_FAC	REAL	1.0	环路输出量的系数
	LMN_OFF	REAL	0.0	环路输出量的偏移量
输出参数	PV	REAL	0.0	标准化的采样值
	ER	REAL	0.0	误差值
	LMN_P	REAL	0.0	比例项
	LMN_I	REAL	0.0	积分项
	LMN_D	REAL	0.0	微分项

（续）

类型	参数名称	数据类型	默认值	说　明
输出参数	QLMN _ HLM	BOOL	FALSE	环路超上限信号
	QLMN _ LLM	BOOL	FALSE	环路超下限信号
	LMN	REAL	0.0	环路小数格式的输出值
	LMN _ PER	WORD	0	环路标准的输出值

2. SFB42/FB42“CONT_ S”步进控制器

SFB42/FB42“CONT_ S”步进控制器可以作为 PI 固定设定值的控制器使用，或作为级联、混料或比例控制器的辅助控制回路中使用，但不能作为主控制器使用。该控制器的功能基于采样控制器的 PI 控制算法，补充了使用模拟激励信号生成二进制（开关）输出信号的功能。当 TI（积分时间常数）= T#0ms 不用积分项时，可以将块作为单纯的比例控制器使用。SFB42/FB42“CONT_ S”步进控制器的控制过程框图如图 8-12 所示。SFB42/FB42“CONT_ S”步进控制器的输入和输出参数如表 8-13 所示。

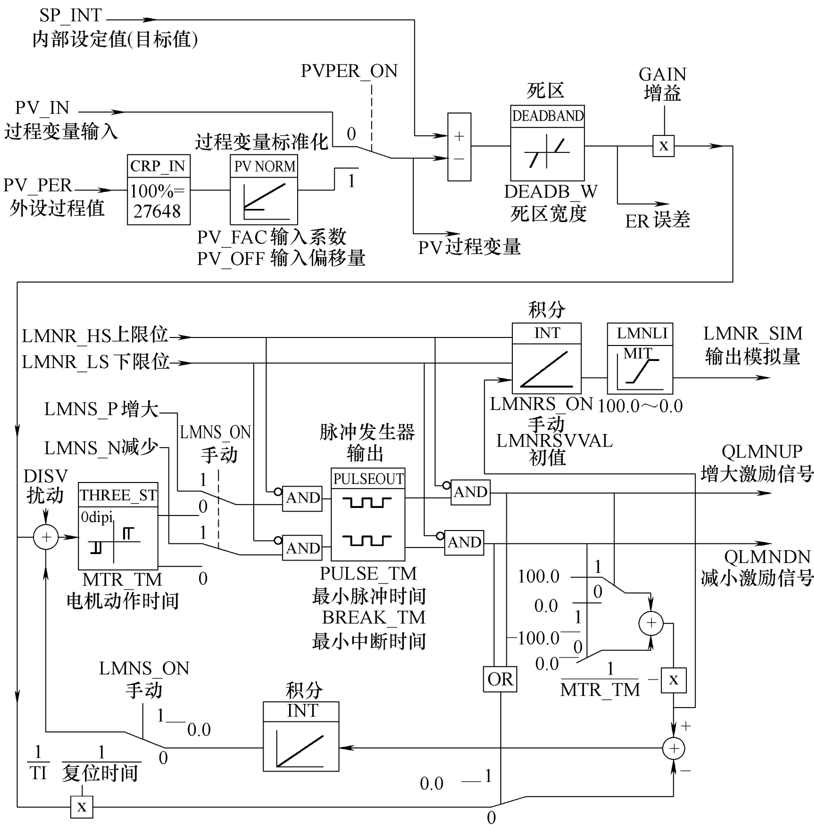


图 8-12　SFB42/FB42“CONT_ S”步进控制器的控制过程框图

表 8-13　SFB42/FB42“CONT_ S”步进控制器输入和输出参数

类型	参数名称	数据类型	默认值	说　明
输入参数	COM _ RST	BOOL	FALSE	完全重新启动，当 =1 时进行初始化
	LMNR _ HS	BOOL	FALSE	=1 表示现在目标在上限位置
	LMNR _ LS	BOOL	FALSE	=1 表示现在目标在下限位置

（续）

类型	参数名称	数据类型	默认值	说　明
输入参数	LMNS _ ON	BOOL	TURE	=1 为手动模式
	LMNUP	BOOL	FALSE	LMNS _ ON =1、LMNUP =1，控制信号增大
	LMNDP	BOOL	FALSE	LMNS _ ON =1、LMNDN =1，控制信号减小
	PVPER _ ON	BOOL	FALSE	=1 选择外围设备信号作为采样值
	CYCLE	TIME	T#Is	采样时间
	SP _ INT	REAL	0.0	环路控制目标值
	PV _ IN	REAL	0.0	小数点采样值
	PV _ PER	WORD	0	16 位整数采样值
	GAIN	REAL	2.0	环路比例增益
	TI	TIME	T#20s	积分时间常数
	DEADB _ W	REAL	1.0	死区宽度
	PV _ FAC	REAL	1.0	PV _ PER 采样值输入系数
	PV _ OFF	REAL	0.0	PV _ PER 采样值偏移量
	PULSE _ TM	TIME	T#3s	最小脉冲时间
	BREAK _ TM	TIME	T#3s	最小中断时间
	MTR _ TM	TIME	T#30s	执行机构从下限（或上限）到上限（或下限）的时间
	DISV	REAL	0.0	扰动输入值
输出参数	QLMNUP	BOOL	FALSE	执行机构增大方向（往上限方向）激励信号
	QLMNDP	BOOL	FALSE	执行机构减小方向（往下限方向）激励信号
	PV	REAL	0.0	标准的采样信号
	ER	REAL	0.0	误差值

PI 步进算法是 SFB42/FB42 在没有真正位置反馈信号的情况下工作。PI 算法的 I 算法和假定的位置反馈信号在同一个积分器（INT）中计算，结果作为反馈值与其余比例算法值进行比较，将误差值应用于三步元素（THREE_ST）和脉冲发生器（PULSEOUT）中。通过调整三步元素的阈值可以降低控制器的切换频率。

脉冲发生器（PULSEOUT）发出的脉冲会受到 PULSE_TM（最小脉冲时间）和 BREAK_TM（最小中断时间）的限制，PULSE_TM 和 BREAK_TM 的具体含义如图 8-13 所示。PULSE_TM 是限制脉冲宽度的最小值，BREAK_TM 是限制两个脉冲的间隔最小值。

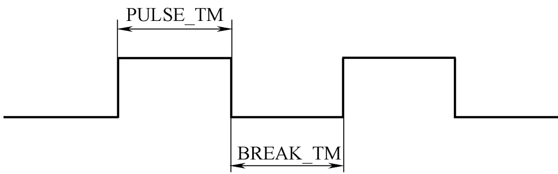


图 8-13　PULSE_ TM 和 BREAK_ TM 的具体含义

正确设置最小脉冲时间或最小中断时间可以防止频繁开/关，频繁开/关会缩短开关元件和执行机构的使用寿命。

使用 LMNR_ HS（上限开关）和 LMNR_ LS（下限开关）分别作为输出脉冲发生器的逻辑控制信号，目的是当 LMNR_ HS =1 时，表示机构达到上限位置，必须停止 QLMNUP（执行机构增大激励信号）输出；当 LMNR_ LS =1 时，表示机构达到下限位置，必须停止 QLMNDN（执行机构减小激励信号）输出，达到极限位置的保护作用。



注意：

1）3 线连接用的公式仅表明了模拟输入模块 SM331（MLFB 号为 6ES7331-7Kxxx-0AB0）的实际测定过程。

2）在 S7-300 系列中，存在一些通过多次测定的模拟输入端。它们规定出公共返回线的线电阻并作数学补偿。所获精确度几乎与 4 线连接可比美。这样模块的一个例子就是 SM331（MLFB 号为 6ES7331-7PF00-0AB0）。

3）所给出的公式仍然适用于主要的物理关系，但并不包含确定 PT100 电阻的有效测定过程。

3. SFB43/FB43 "PULSEGEN" 脉冲发生器

SFB43/FB43 "PULSEGEN" 脉冲发生器用于构建具有比例执行机构脉冲输出的 PID 控制器，通常与 SFB41/FB41 "CONT_ C" 连续控制器结合使用，如图 8-14 所示，可以组态具有脉宽调制功能的 PID 两步/三步控制器。

SFB43/FB43 "PULSEGEN" 脉冲发生器的输入及输出参数如表 8-14 所示。

表 8-14　SFB43/FB43 "PULSEGEN" 脉冲发生器的输入及输出参数

类型	参数名称	数据类型	默认值	说　明
输入参数	INV	REAL	0.0	"输入变量",该参数一般取模拟操作值
	PER _ TM	TILME	T#Is	脉宽调制的恒定周期值,该参数对应于控制器的采样时间,脉冲发生器采样时间与控制器采样时间的比率决定脉宽调制的精度
	P _ B _ TM	TIME	T#0s	可以在输入参数"最小脉冲或最小中断时间"分配最小脉冲或最小中断时间
	RATIOFAC	REAL	1.0	"比率因子",该参数可以改变负脉冲宽度与正脉冲宽度的比例
	STEP3 _ ON	BOOL	TRUE	"启用三步控制",STEP3 _ ON =1,启用三步控制,两个输出信号都处于激活状态
	ST2BA _ ON	BOOL	FALSE	"启用双极操作值范围的两步控制",当 STEP3 _ ON =0 时,可以在"双极操作值的两步控制"模式和"单极操作值范围的两步控制"模式间互相切换
	MAN _ ON	BOOL	FALSE	"启用手动模式",当 MAN _ ON =1 时,可以手动设置输出信号
输入参数	POS _ P _ ON	BOOL	FALSE	"启用正脉冲",在手动模式及 POS _ P _ ON =1 下: 采用三步控制时,输出信号是 QPOS _ P; 采用两步控制时,输出信号 QNEG _ P 始终为与 QPOS _ P 反向
	NEG _ P _ ON	BOOL	FALSE	"启用负脉冲",在手动模式及 NEG _ P _ ON =1 下: 采用三步控制时,输出信号是 QNEG _ P: 采用两步控制时,输出信号 QNEG _ P 始终为与 QPOS _ P 反向
	SYN _ ON	BOOL	TRUE	"启用同步",当 SYN _ ON =1 时,自动与更新输入变量 INV 的块同步
	COM _ RST	BOOL	FALSE	当 COM _ RST =1 时,自动执行初始化程序

（续）

类型	参数名称	数据类型	默认值	说　明
输入参数	CYCLE	TIME	T#10ms	“采样时间”,块调用的时间,必须为“常数”。建议设置与输入变量 INV 的更新周期相同
输出参数	QPOS _ P	BOOL	FALSE	“输出正脉冲”: 在三步控制中,始终为正脉中 在两步控制中,QNEG _ P 始终与 QPOS _ P 反向
	QNEG _ P	BOOL	FALSE	“输出负脉冲” 在三步控制中,始终为负脉冲 在两步控制中,QNEG _ P 始终与 QPOS _ P 反向

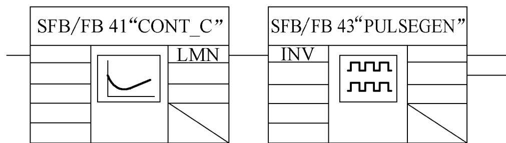


图 8-14　“PULSEGEN”和“CONT_ C”结合使用

SFB43/FB43“PULSEGEN”脉冲发生器具体的功能是将来自 SFB41/FB41“CONT_ C”连续控制器的输出变量作为输入变量，通过脉宽调制将输入变量 INV（= SFB41/FB41“CONT_ C”控制器的输出值）转换为具有恒定周期的脉冲列，每周期脉冲宽度与输入变量成比例。这里需要使 SFB41/FB41“CONT_ C”控制器的输出值更新周期与 SFB43/FB43“PULSEGEN”脉冲发生器的采样周期相同。

分配给 PER_ TM 的周期与 SFB43/FB43“PULSEGEN”的处理周期不完全相同。PER_ TM 周期是输出脉冲的恒定周期，PER_ TM 周期由若干个 SFB43/FB43“PULSEGEN”处理周期组成，因此可以将每个 PER_ TM 周期 SFB43/FB43“PULSEGEN”调用的次数作为脉宽调制精度的衡量标准。PER_ TM 的周期与 SFB43/FB43“PULSEGEN”的处理周期的关系如图 8-15 所示，图中以 QPOS_ P 输出为例。

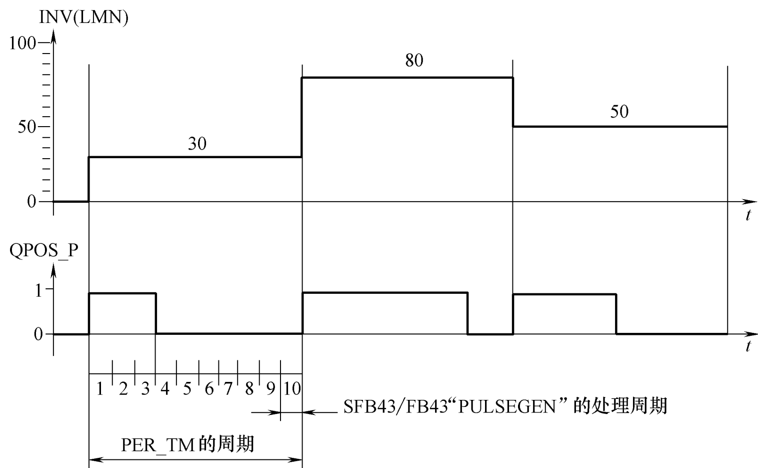


图 8-15　PER_ TM 的周期与 SGB43/FB43“PULSEGEN”处理周期的关系

1）脉宽调制（脉冲宽度调制）。在图 8-15 中，假定一个 PER_ TM 周期里调用 10 个

“PULSEGEN” 的处理周期，那么当输入变量 INV 为 30% 的最大值时，则调用 “PULSEGEN” “10×30%” 个 PER_ TM 周期的时间，输出 QPOS_ P 为 “1”，其余时间 “10×70%” 输出 QPOS_ P 为 “0”。

2）调制精度。在图 8-15 中，在 PER_ TM 一个周期里调用 “PULSEGEN” 的处理次数的比例就是反应控制精度的比例。如在图 8-15 中，一个 PER_ TM 周期里调用 10 个 “PULSEGEN” 的处理周期，那么控制精度为 1:10，换句话说就是输入值 INV 只能映射 10% 为量化单位的 QPOS_ P 脉冲输出占空比。

3）自动同步　可以将脉冲输出与更新输入变量 INV（例如，CONT_ C）的块同步。这样可以保证输入变量的变化及时反应到输出脉冲中。

脉冲发生器以 PER_ TM 周期采样输入值 INV，并将该值转换为相应长度的脉冲信号。但是，由于 INV 常是在较慢的周期性中断级别计算，脉冲发生器应在 INV 更新后尽快开始将离散值转换为脉冲信号。因此，需要采用同步技术。

如果 INV 发生了变化，且 INV 采样不是发生在 PER_ TM 周期中 “PULSEGEN” 处理的前或后两个调用周期的区间里，则执行同步，并重新计算脉冲宽度，在下一周期输出新的脉冲；如果 INV 发生了变化，且 INV 采样是发生在 PER_ TM 周期中 “PULSEGEN” 处理的前或后两个调用周期的区间里，则不需要执行同步。自动同步示意图如图 8-16 所示。如果使 “SYN_ ON” = FALSE，则禁用自动同步。

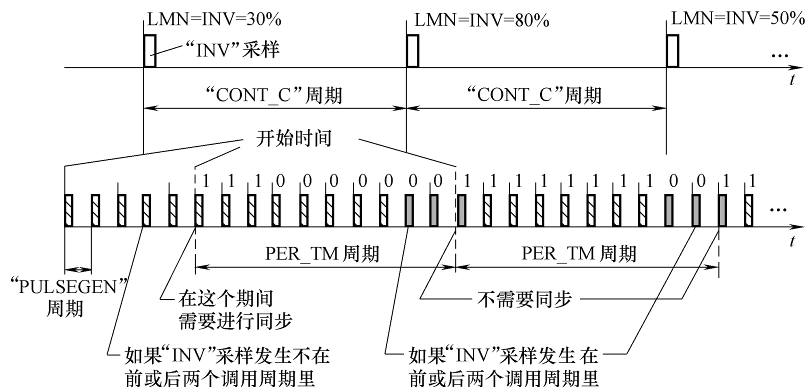


图 8-16　自动同步示意图

SFB43/FB43 “PULSEGEN” 脉冲发生器可以组态为三步输出、双极或单极的两步输出的 PID 控制器。表 8-15 列出了可能模式的开关组合参数设置。

表 8-15　SFB43/FB43 “PULSEGEN” 脉冲发生器可能模式的开关组合

模式			MAN _ ON	STEP3 _ ON	ST2BI _ ON
自动	三步控制	三步控制	0	1	0/1
	两步控制	双极性(-100% ~ +100%)	0	0	1
		单极性(0% ~100%)	0	0	0
手动模式			1	0/1	0/1

在手动模式（MAN_ ON = 1）下，无论 INV 为何值，均可使用信号 POS_ P_ ON 和 NEG_ P_ ON 设置三步或两步控制器的开关量输出。两步/三步控制的手动模式输入与输出如表 8-16 所示。

表 8-16　两步/三步控制的手动模式下的输入与输出

手动模式下	POS _ P _ ON	NEG _ P _ ON	QPOS _ P	QNEG _ P
三步控制	0	0	0	0
	1	0	1	0
	0	1	0	1
	1	1	0	0
二步控制	0	0/1	0	1
	1	0/1	1	0

4. FB58 温度连续控制器及 FB58 温度步进控制器

FB58“TCONT _ CP”用于使用连续或脉冲控制信号来控制温度过程。可以设置参数，启用或禁止 PID 控制器的子功能，以便使其和要控制的过程相适应。使用参数分配工具可以很简单地进行这些设置。

模块功能以 PID 控制算法为基础，带有用于温度过程的附加功能。控制器提供了模拟量调节值和脉宽调制驱动信号。控制器将信号输出到一个执行器；换句话说，通过一个控制器，可以加热，也可以冷却，但是不能同时加热和冷却。FB58“TCONT _ CP”的控制过程框图如图 8-17 所示。

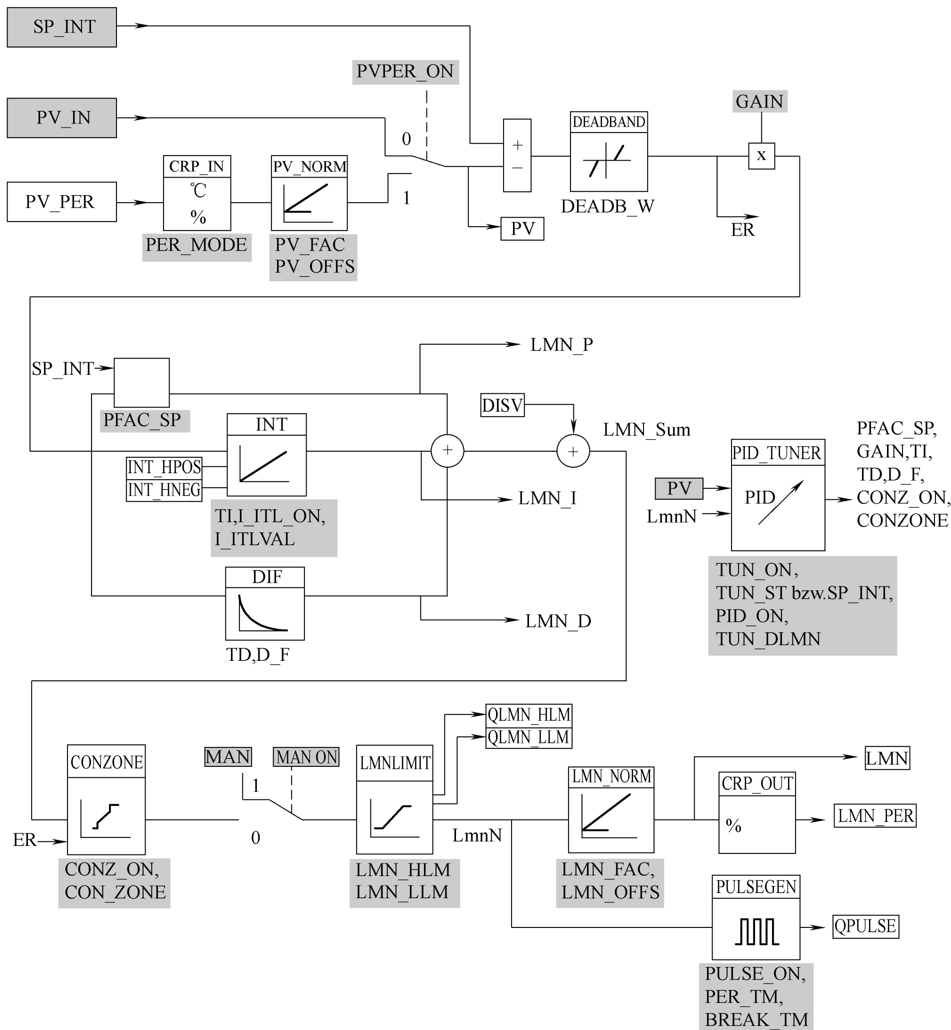


图 8-17　FB58“TCONT _ CP”温度步进控制器的控制过程框图

FB58 “TCONT _ CP” 的输入及输出参数如表 8-17 所示。

表 8-17　FB58 “TCONT _ CP” 脉冲发生器的输入及输出参数

类型	参数名称	数据格式	默认值	说　明
输入	PB _ IN	REAL	0.0	实数类型过程值输入
输入	PV _ PER	INT	0	整数类型过程值输入
输入	DISV	REAL	0.0	干扰补偿
输入	INT _ HPOS	BOOL	FALSE	正向积分保持开启
输入	INT _ HNEG	BOOL	FALSE	反向积分保持开启
输入	SELECT	INT	0	PID 功能和脉冲功能的选择
输入	PV	REAL	0.0	过程值
输出	LMN	REAL	0.0	实数类型的 PID 输出控制量
输出	LMN _ PER	INT	0	整数类型的 PID 输出控制量
输出	QPULSE	BOOL	FALSE	输出脉冲信号
输出	QLMN _ HLM	BOOL	FALSE	控制量到达上限
输出	QLMN _ LLM	BOOL	FALSE	控制量到达下限
输出	QC _ ACT	BOOL	TRUE	下一周期 PID 是否执行
输入_输出	CYCLE	REAL	0.1	PID 计算的周期时间
输入_输出	CYCLE _ P	REAL	0.02	脉冲输出的刷新时间
输入_输出	SP _ INP	REAL	0	设定值
输入_输出	MAN	REAL	0	手动值
输入_输出	COM _ RST	BOOL	FALSE	复位
输入_输出	MAN _ ON	BOOL	TRUE	手/自动,默认为手动
静态变量	DEADB _ W	REAL	0.0	偏差死区
静态变量	I _ ITLVAL	REAL	0.0	初始积分值
静态变量	LMN _ HLM	REAL	100.0	控制量上限
静态变量	LMN _ LLM	REAL	0.0	控制量下限
静态变量	PV _ FAC	REAL	1.0	过程值转换因子
静态变量	PV _ OFFS	REAL	0.0	过程值转换偏移量
静态变量	LMN _ FAC	REAL	1.0	输出控制量转换因子
静态变量	LMN _ OFFS	REAL	0.0	输出控制量转换偏移量
静态变量	PER _ TM	REAL	1.0	脉冲输出的周期时间
静态变量	P _ B _ TM	REAL	0.0	最小脉冲高电平/低电平时间
静态变量	TUN _ DLMN	REAL	20.0	整定中的控制输出变化量
静态变量	PER _ MODE	INT	0	整数类型输入转换模式
静态变量	PVPER _ ON	BOOL	FALSE	输入过程值通道选择
静态变量	I _ ITL _ ON	BOOL	FALSE	积分功能初始化
静态变量	PULSE _ ON	BOOL	FALSE	脉冲输出使能
静态变量	ER	REAL	0.0	偏差信号

（续）

类型	参数名称	数据格式	默认值	说　明
静态变量	LMN _ P	REAL	0. 0	比例项结果
静态变量	LMN _ I	REAL	0. 0	积分项结果
静态变量	LMN _ D	REAL	0. 0	微分项结果
静态变量	PHASE	INT	0	自整定步骤
静态变量	STATUS _ H	INT	0	自整定状态值
静态变量	STATUS _ D	INT	0	过程类型
静态变量	QTUN _ RUN	BOOL	FALSE	阶段 2 已激活(整定中)
静态变量	PI _ CON. GAIN	REAL	0. 0	PI _ CON 中的比例参数
静态变量	PI _ CON. TI	REAL	0. 0	PI _ CON 中的积分参数
静态变量	PID _ CON. GAIN	REAL	0. 0	PID _ CON 中的比例参数
静态变量	PID _ CON. TI	REAL	0. 0	PID _ CON 中的积分参数
静态变量	PID _ CON. TD	REAL	0. 0	PID _ CON 中的微分参数
静态变量	PAR _ SAVE. PFAC _ SP	REAL	1. 0	PAR _ SAVE 中的比例弱化因子
静态变量	PAR _ SAVE. GAIN	REAL	0. 0	PAR _ SVE 中的比例参数
静态变量	PAR _ SAVE. TI	REAL	0. 0	PAR _ SAVE 中的积分参数
静态变量	PAR _ SAVE. TD	REAL	0. 0	PAR _ SAVE 中的微分参数
静态变量	PAR _ SAVE. D _ F	REAL	5. 0	PAR _ SVE 中的微分因子
静态变量	PAR _ SAVE. CON _ ZONE	REAL	100. 0	PAR _ SAVE 中的控制带
静态变量	PAR _ SAVE. CONZ _ ON	BOOL	FALSE	PAR _ SAVE 中的控制带使能
静态变量	PFAC _ SP	REAL	1. 0	设定值改变时的比例弱化因子
静态变量	GAIN	REAL	2. 0	比例增益
静态变量	TI	REAL	10. 0	积分时间
静态变量	TD	REAL	10. 0	微分时间
静态变量	D _ F	REAL	5. 0	微分因子
静态变量	CON _ ZONE	REAL	100. 0	控制带
静态变量	CONZ _ ON	BOOL	FALSE	控制带使能
静态变量	TUN _ ON	BOOL	FALSE	自整定使能
静态变量	UNDO _ PAR	BOOL	FALSE	参数恢复使能
静态变量	SAVE _ PAR	BOOL	FALSE	参数保存使能
静态变量	LOAD _ PID	BOOL	FALSE	装载 PID/PI _ CON 参数
静态变量	PID _ ON	BOOL	TRUE	PID 或 PI 选择

附　录

附录 A　S7 指令速查表

附表 A-1　S7 指令速查表

助记符	程序元素分类	功 能 说 明
+	整数算术运算指令	加上一个整数常数(16 位,32 位)
=	位逻辑指令	赋值
)	位逻辑指令	嵌套闭合
+ AR1	累加器指令	AR1 加累加器 1 到地址寄存器 1
+ AR2	累加器指令	AR1 加累加器 1 到地址寄存器 2
+ D	整数算术运算指令	作为 32 位双整数,将累加器 1 和累加器 2 中的内容相加
- D	整数算术运算指令	作为 32 位双整数,将累加器 2 中的内容减去累加器 1 中的内容
* D	整数算术运算指令	作为 32 位双整数,将累加器 1 和累加器 2 中的内容相乘
/D	整数算术运算指令	作为 32 位双整数,将累加器 2 中的内容除以累加器 1 中的内容
? D	比较指令	两个 32 位双整数比较:= =,< >,>,<,> =,< =
+ I	整数算术运算指令	作为 16 位整数,将累加器 1 和累加器 2 中的内容相加
- I	整数算术运算指令	作为 16 位整数,将累加器 2 中的内容减去累加器 1 中的内容
* I	整数算术运算指令	作为 16 位整数,将累加器 1 和累加器 2 中的内容相乘
/I	整数算术运算指令	作为 16 位整数,将累加器 2 中的内容除以累加器 1 中的内容
? I	比较指令	两个 16 位整数比较:= =,< >,>,<,> =,< =
+ R	浮点数算术运算指令	作为 32 位浮点数(IEEE-FP),将累加器 1 和累加器 2 中的内容相加
- R	浮点数算术运算指令	作为 32 位浮点数(IEEE-FP),将累加器 2 中的内容减去累加器 1 中的内容
* R	浮点数算术运算指令	作为 32 位浮点数(IEEE-FP),将累加器 1 和累加器 2 中的内容相乘
/R	浮点数算术运算指令	作为 32 位浮点数(IEEE-FP),将累加器 2 中的内容除以累加器 1 中的内容
A	位逻辑指令	逻辑“与”操作
A(	位逻辑指令	逻辑“与”操作嵌套开始
ABS	浮点算术运算指令	浮点数取绝对值(32 位,IEEE-FP)
ACOS	浮点算术运算指令	浮点数反余弦运算(32 位,IEEE-FP)
AD	字逻辑指令	32 位双字逻辑“与”操作
AN	位逻辑指令	位逻辑“与非”操作
AN(	位逻辑指令	位逻辑“与非”操作嵌套开始
ASIN	浮点算术运算指令	浮点数反正弦运算(32 位,IEEE-FP)
ATAN	浮点算术运算指令	浮点数反正切运算(32 位,IEEE-FP)

（续）

助记符	程序元素分类	功 能 说 明
AW	字逻辑指令	16 位字逻辑“与”操作
BE	程序控制指令	块结束
BEC	程序控制指令	条件块结束
BEU	程序控制指令	无条件块结束
BLD	程序控制指令	程序显示指令(空)
BTD	转换指令	BCD 转成 32 位双整数
BTI	转换指令	BCD 转成 16 位整数
CAD	转换指令	在累加器 1 中变换字节序列(32 位)
CALL	程序控制指令	调用多背景块
		从库中调用块
CAW	转换指令	在累加器 1 中变换字节序列(16 位)
CC	程序控制指令	条件调用
CD	计数器指令	减计数器
CDB	转换指令	交换共享数据块和背景数据块
CLR	位逻辑指令	RLO 清零(RLO =0)
COS	浮点算术运算指令	浮点数余弦运算(32 位,IEEE-FP)
CU	计数器指令	加计数器
DEC	累加器指令	减少累加器 1 低字的低字节
DTB	转换指令	32 位双整数转成 BCD 码
DTR	转换指令	32 位双整数转成浮点数(32 位,IEEE-FP)
ENT	累加器指令	进入累加器栈
EXP	浮点算术运算指令	浮点数指数运算(32 位,IEEE-FP)
FN	位逻辑指令	脉冲下降沿
FP	位逻辑指令	脉冲上升沿
FR	计数器指令	使能计数器(FRC0 ~ C255)
	定时器指令	使能定时器(任意)
INC	累加器指令	增加累加器 1 低字的低字节
INVD	转换指令	对 32 位双整数求反码
INVI	转换指令	对 16 位整数求反码
IYB	转换指令	16 位整数转换成 BCD
ITD	转换指令	16 位整数转换成 32 位双整数
JBI	跳转指令	若 BR =1,则跳转
JC	跳转指令	若 RLO =1,则跳转
JCB	跳转指令	若 RLO =1,且 BR =1,则跳转
JCN	跳转指令	若 RLO =0,则跳转
JL	跳转指令	跳转到标号
JM	跳转指令	若负,则跳转
JMZ	跳转指令	若负或零,则跳转
JN	跳转指令	若非零,则跳转
JNB	跳转指令	若 RLO =0,且 BR =1,则跳转
JNBI	跳转指令	若 BR =0,则跳转
JO	跳转指令	若 OV =1,则跳转
JOB	跳转指令	若 OS =1,则跳转
JP	跳转指令	若正,则跳转
JPZ	跳转指令	若正或零,则跳转
JU	跳转指令	无条件跳转
L	装入/传送指令	装入
L DBLG	装入/传送指令	将共享数据块的长度装入累加器 1 中

（续）

助记符	程序元素分类	功 能 说 明
L DBNO	装入/传送指令	将共享数据块的块号装入累加器 1 中
L DILG	装入/传送指令	将背景数据块的长度装入累加器 1 中
L DINO	装入/传送指令	将背景数据块的块号装入累加器 1 中
L STW	装入/传送指令	将状态字装入累加器 1
L	定时器指令	将当前定时值作为整数装入累加器 1(当前定时值可以是 0～255 的一个数字,例如 L T12)
	计数器指令	将当前计数值作为整数装入累加器 1(当前计数值可以是 0～255 的一个数字,例如 L C6)
LAR1	装入/传送指令	将累加器 1 中的内容装入地址寄存器 1
LAR1 <D>	装入/传送指令	将两个双整数(32 位指针)装入地址寄存器 1
LAR1 AR2	装入/传送指令	将地址寄存器 2 的内容装入地址寄存器 1
LAR2	装入/传送指令	将累加器 2 中的内容装入地址寄存器 1
LAR2 <D>	装入/传送指令	将两个双整数(32 位指针)装入地址寄存器 2
LC	计数器指令	将当前计数值作为 BCD 码装入累加器 1(当前计数值可以是 0～255 的一个数字,例如 LC C10)
	定时器指令	将当前定时值作为 BCD 码装入累加器 1(当前定时值可以是 0～255 的一个数字,例如 LC T13)
LEAVE	累加器指令	离开累加器栈
LN	浮点算术运算指令	浮点数自然对数运算(32 位)
LOOP	跳转指令	循环
MCR(	程序控制指令	将 RLO 存入 MCR 堆栈,开始 MCR
)MCR	程序控制指令	结束 MCR
MCRA	程序控制指令	激活 MCR 区域
MCRD	程序控制指令	取消激活 MCR 区域
MOD	整数算术运算指令	32 位双整数形式的除法,其结果为余数
NEGD	转换指令	对 32 位双整数求补码
NEGI	转换指令	对 16 位整数求补码
NEGR	转换指令	对浮点数求反码(32 位,IEEE-FP)
NOP 0	累加器指令	空指令
NOP 1	累加器指令	空指令
NOT	位逻辑指令	RLO 取反
O	位逻辑指令	逻辑“或”操作
O(	位逻辑指令	逻辑“或”操作嵌套开始
OD	字逻辑指令	32 位双字逻辑“或”操作
ON	位逻辑指令	逻辑“或非”操作
ON(	位逻辑指令	逻辑“或非”操作嵌套开始
OPN	数据块调用指令	打开数据块
OW	字逻辑指令	16 位字逻辑“或”操作
POP	累加器指令	带有 2 个累加器的 CPU,把 ACCU2 弹出 ACCU1 指令
		带有 4 个累加器的 CPU,把 ACCU2 弹出 ACCU1 指令
PUSH	累加器指令	带有 2 个累加器的 CPU,ACCU1 推入 ACCU2 指令
		带有 4 个累加器的 CPU,ACCU1 推入 ACCU2 指令
R	位逻辑指令	复位
	计数器指令	复位计数器(当前计数值可以是 0～255 的一个数字,例如 R C3)
	定时器指令	复位定时器(当前定时值可以是 0～255 的一个数字,例如 R T10)

（续）

助记符	程序元素分类	功 能 说 明
RLD	移位和循环移位指令	32 位双字循环左移
RLDA	移位和循环移位指令	通过 CC1 累加器 1 循环左移(32 位)
RND	转换指令	取整
RND –	转换指令	向下舍入为双整数
RND +	转换指令	向上舍入为双整数
RRD	移位和循环移位指令	双字循环右移(32 位)
RRDA	移位和循环移位指令	通过 CC1 累加器 1 循环右移(32 位)
S	位逻辑指令	置位
	计数器指令	置位计数器(当前计数值可以是 0 ~255 的一个数字,例如 S C12)
SAVE	位逻辑指令	把 RLO 存入 BR 寄存器
SD	定时器指令	延时接通定时器
SE	定时器指令	延时脉冲定时器
SET	位逻辑指令	置位
SF	定时器指令	延时断开定时器
SIN	浮点算术运算指令	浮点数正弦运算(32 位)
SLD	移位和循环移位指令	32 位双字左移
SLW	移位和循环移位指令	16 位字左移
SP	定时器指令	脉冲定时器
SQR	浮点算术运算指令	浮点数平方运算(32 位)
SQRT	浮点算术运算指令	浮点数平方根运算(32 位)
SRD	移位和循环移位指令	32 位双字右移
SRW	移位和循环移位指令	16 位字右移
SS	定时器指令	保持型延时接通定时器
SSD	移位和循环移位指令	32 位有符号双整数移位
SSI	移位和循环移位指令	16 位有符号整数移位
T	装入/传送指令	传送
T STW	装入/传送指令	将累加器 1 中的内容传送到状态字
TAK	累加器指令	累加器 1 与累加器 2 进行互换
TAN	浮点算术运算指令	32 位浮点数正切运算
TAR1	装入/传送指令	将地址寄存器 1 中的内容传送到累加器 1
		将地址寄存器 1 中的内容传送到目的地(32 位指针)
		将地址寄存器 1 中的内容传送到地址寄存器 2
TAR2	装入/传送指令	将地址寄存器 2 中的内容传送到累加器 1
		将地址寄存器 2 中的内容传送到目的地(32 位指针)
TRUNC	转换指令	截尾取整
UC	程序控制指令	无条件调用
X	位逻辑指令	逻辑“异或”操作
X(	位逻辑指令	逻辑“异或”操作嵌套开始
XN	位逻辑指令	逻辑“异或非”操作
XN(	位逻辑指令	逻辑“异或非”操作嵌套开始
XOD	位逻辑指令	32 位双字逻辑“异或”操作
XOW	位逻辑指令	16 位字逻辑“异或”操作

附表 A-2　S7-300/400 LAD 指令速查表

助记符	程序元素分类	功 能 说 明
--|　|--	位逻辑指令	常开触点(地址)
--|/|--	位逻辑指令	常闭触点(地址)
--[ ]	位逻辑指令	输出线圈
--[#]--	位逻辑指令	中间输出
==0--||--	状态位指令	结果位等于“0”
>0--||--	状态位指令	结果位大于“0”
> =0--||--	状态位指令	结果位大于等于“0”
< =0--||--	状态位指令	结果位小于等于“0”
<0--||--	状态位指令	结果位小于“0”
< >0--||--	状态位指令	结果位不等于“0”
ABS	浮点数算术运算指令	浮点数绝对值运算
ACOS	浮点数算术运算指令	浮点数反余弦运算
ADD _ DI	整数算术运算指令	双整数加法
ADD _ I	整数算术运算指令	整数加法
ADD _ R	浮点数算术运算指令	浮点数加法
ASIN	浮点数算术运算指令	浮点数反正弦运算
ATAN	浮点数算术运算指令	浮点数反正切运算
BCD _ DI	转换指令	BCD 码转换为双整数
BCD _ I	转换指令	BCD 码转换为整数
BR--||--	状态位指令	异常位二进制结果
--(CALL)	程序控制指令	从线圈调用 FC/SF(无参数)
CALL _ FB	程序控制指令	从方块调用 FB
CALL _ FC	程序控制指令	从方块调用 FC
CALL _ SFB	程序控制指令	从方块调用 SFB
CALL _ SFC	程序控制指令	从方块调用 SFC
--(CD)	计数器指令	减计数器线圈
CEIL	转换指令	上取整
CMP > =D	比较指令	双整数比较(= = 、< > 、> 、< 、> = 、< =)
CMP > =I	比较指令	整数比较(= = 、< > 、> 、< 、> = 、< =)
CMP > =R	比较指令	浮点数比较(= = , < > 、> 、< 、> = 、< =)
COS	浮点数算术运算指令	浮点数余弦运算
--(CU)	计数器指令	加计数器线圈
DI _ BCD	转换指令	双整数转换为 BCD 码
DI _ R	转换指令	双整数转换为浮点数
DIV _ DI	整数算术运算指令	双整数除法
DIV _ I	整数算术运算指令	整数除法
DIV _ R	浮点数算术运算指令	浮点数除法
EXP	浮点数算术运算指令	浮点数指数运算
FLOOR	转换指令	下取整
I _ BCD	转换指令	整数转换为 BCD 码
I _ DI	转换指令	整数转换为双整数
INV _ I	转换指令	整数的二进制反码
INV _ DI	转换指令	双整数的二进制反码
--(TMP)	跳转指令	无条件跳转
--(JMP)	跳转指令	条件跳转
--(JMPN)	跳转指令	若非则跳转
LABEL	跳转指令	标号
LN	浮点数算术运算指令	浮点数自然对数运算

（续）

助记符	程序元素分类	功 能 说 明
--(MCR>)	程序控制指令	主控继电器断开
--(MCR<)	程序控制指令	主控继电器接通
--(MCRA)	程序控制指令	主控继电器启动
--(MCRD)	程序控制指令	主控继电器停止
MOD_DI	整数算术运算指令	回送余数的双整数
MOVE	赋值指令	赋值
MUL_DI	整数算术运算指令	双整数乘法
MUL_I	整数算术运算指令	整数乘法
MUL_R	浮点数算术运算指令	浮点数乘法
--(N)--	位逻辑指令	RLO 下降沿检测
NEG	位逻辑指令	地址下降沿检测
NEG_DI	转换指令	双整数的二进制补码
NEG_I	转换指令	整数的二进制补码
NEG_R	转换指令	浮点数求反
--|NOT|--	位逻辑指令	信号流反向
--(OPN)	数据块调用指令	打开数据块：DB 或 DI
OS--| |--	状态位指令	存储溢出异常位
OV--| |--	状态位指令	溢出异常位
--(P)--	位逻辑指令	RLO 上升沿检测
POS	位逻辑指令	地址上升沿检测
--(R)	位逻辑指令	线圈复位
--(RET)	程序控制指令	返回
ROL_DW	移位和循环指令	双字左循环
ROR_DW	移位和循环指令	双字右循环
ROUND	转换指令	舍入为双整
RS	位逻辑指令	复位置位触发器
--(S)	位逻辑指令	线圈置位
--(SAVE)	位逻辑指令	将 RLO 存入 BR 存储器
--(SC)	计数器指令	设置计数器值
S_CD	计数器指令	减计数器
S_CU	计数器指令	加计数器
S_CUD	计数器指令	加减计数器
--(SD)	定时器指令	接通延时定时器线圈
--(SE)	定时器指令	扩展脉冲定时器线圈
--(SF)	定时器指令	断开延时定时器线圈
SHL_DW	移位和循环指令	双字左移
SHL_W	移位和循环指令	字左移
SHR_DI	移位和循环指令	双整数右移
SHR_DW	移位和循环指令	双字右移
SHR_I	移位和循环指令	整数右移
SHR_W	移位和循环指令	字右移
SIN	浮点数算术运算指令	浮点数正弦运算
S_ODT	定时器指令	接通延时 S5 定时器
S_ODTS	定时器指令	保持型接通延时 S5 定时器
S_OFFDT	定时器指令	断电延时 S5 定时器
--(SP)	定时器指令	脉冲延时定时器线圈
S_PEXT	定时器指令	扩展脉冲 S5 定时器
S_PULSE	定时器指令	脉冲 S5 定时器

（续）

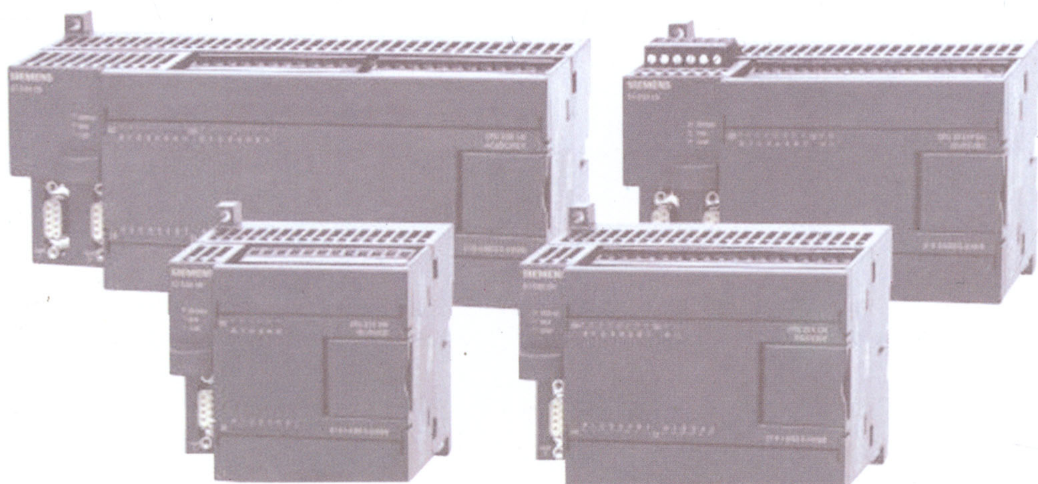
助记符	程序元素分类	功 能 说 明
SQR	浮点数算术运算指令	浮点数平方
SQRT	浮点数算术运算指令	浮点数平方根
SR	位逻辑指令	置位复位触发器
--(SS)	定时器指令	保持型接通延时定时器线圈
SUB _ DI	整数算术运算指令	双整数减法
SUB _ I	整数算术运算指令	整数减法
SUB _ R	整数算术运算指令	浮点数减法
TAN	浮点数算术运算指令	浮点数正切运算
TRUNC	转换指令	舍去小数取整为双整数
U0--||--	状态位指令	无序异常位
WAND _ DW	字逻辑指令	双字和双字相“与”
WAND _ W	字逻辑指令	字和字相“与”
WOR _ DW	字逻辑指令	双字和双字相“或”
WOR _ W	字逻辑指令	字和字相“或”
WXOR _ DW	字逻辑指令	双字和双字相“异或”
WXOR _ W	字逻辑指令	字和字相“异或”

附录 B　梯形图指令速查表

助记符	说　明	助记符	说　明
位逻辑指令			
—|　|	动合触点(地址)	—|/|	动断触点(地址)
—( )	输出线圈	—(#)	中间输出
—(N)—	RLO 负跳沿检测	NEG	地址下降沿检测
—|NOT|—	取反能流	—(P)—	RLO 正跳沿检测
POS	地址上升沿检测	—(R)	复位线圈
RS	复位置位触发器	—(S)	置位线圈
—(SAVE)	将 RLO 的状态保存到 BR	SR	置位复位触发器
==0—||—	结果位等于 0	>0—||—	结果位大于 0
<0—||—	结果位小于 0	> =—||—	结果位大于等于 0
< =0—||—	结果位小于等于 0	< >0—||—	结果位不等于 0
BR—||—	异常位二进制结果	OS—||—	存储的异常位溢出
OV—||—	异常位溢出	UO—||—	异常位无序
CMP > =D	比较双精度整数(= = 、< > 、> 、< 、> = 、< =)	CMP > =I	比较整数(==、< > 、> 、< 、> = 、< =)
CMP > =R	比较实数(==、< > 、> 、< 、> = 、< =)		
计数器、定时器指令			
—(CD)	减计数器线圈	—(CU)	加计数器线圈
—(SC)	设置计数器值	S _ CD	减计数器
S _ CU	加计数器	S _ CUD	双向计数器
—(SD)	接通延时定时器线圈	—(SE)	扩展脉冲定时器线圈
—(SF)	断开延时定时器线圈	—(SP)	脉冲定时器线圈
S _ ODTS	保持接通延时 S5 定时器	—(SS)	保持接通延时定时器线圈
S _ ODT	接通延时 S5 定时器	S _ PEXT	扩展脉冲 S5 定时器
S _ PULSE	脉冲 S5 定时器	S _ OFFDT	断开延时 S5 定时器

（续）

数字指令			
BCD_DI	BCD 码转换为双精度整数	WXOR_DW	异或运算双字
CEIL	上限	ROL_DW	循环左移双字
DI_R	双精度整数转换为浮点数	SHL_DW	左移双字
I_BCD	整数转换为 BCD 码	SHR_DW	右移双字
INV_I	二进制反码整数	SHR_DI	右移双精度整数
BCD_I	BCD 码转换为整数	NEG_I	二进制补码整数
DI_BCD	双精度整数转换为 BCD 码	ROUND	取整为双精度整数
FLOOR	基数	MOVE	数据传送
I_DI	整数转换为双精度整数	ADD_I	加整数
INV_DI	二进制反码双精度整数	DIV_I	除整数
NEG_DI	二进制补码双精度整数	MUL_I	乘整数
NEG_R	取反浮点数数字	SUB_I	减整数
TRUNC	截断双精度整数部分	ABS	取浮点数数字的绝对值
ADD_DI	加双精度整数	ACOS	取反余弦值
DIV_DI	除双精度整数	SIN	取正弦值
MUL_DI	乘双精度整数	TAN	取正切值
SUB_DI	减双精度整数	SUB_R	减实数
MOD_DI	返回分数双精度整数	DIV_R	除实数
ASIN	取反正弦值	LN	得到自然对数
ATAN	取反正切值	SQRT	取平方根
COS	取余弦值	WAND_W	与运算字
ADD_R	加实数	WOR_W	或运算字
MUL_R	乘实数	WXOR_W	异或运算字
EXP	得到指数值	ROR_DW	循环右移双字
SQR	取平方值	SHL_W	左移字
WAND_DW	与运算双字	SHR_W	右移字
WOR_DW	或运算双字	SHR_I	右移整数
控制指令			
—(CALL)	调用来自线圈的 FC SFC(不带参数)	CALL_FB	从逻辑框中调用 FB
CAQLL_FC	从逻辑框中调用 FC	CALL_SFB	从逻辑框中调用系统 FB
CALL_SFC	从逻辑框中调用系统 FC	—(JMP)	无条件/有条件跳转
—(JMPN)	如果非则跳转	LABEL	标号
—(MCR >)	主控制断电器关闭	—(MCR <)	主控制断电器开启
—(MCRA)	主控制断电器激活	—(MCRD)	主控制断电器取消激活
—(RET)	返回		



地址：北京市百万庄大街22号
邮政编码：100037
电话服务
社服务中心：010-88361066
销售一部：010-68326294
销售二部：010-88379649
读者购书热线：010-88379203
网络服务
教材网：http://www.cmpedu.com
机工官网：http://www.cmpbook.com
机工官博：http://weibo.com/cmp1952
封面无防伪标均为盗版

上架指导 工业技术/电气自动化/PLC

ISBN 978-7-111-44821-1

策划编辑◎朱林

ISBN 978-7-111-44821-1



9 787111 448211 >

定价：39.80元